DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (*if applicable*): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (*City, State, and ZIP Code*): Monterey, CA 93943-5000	7b. ADDRESS (*City, State, and ZIP Code*): Monterey, CA 93943-5000
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (*if applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (*City, State, and ZIP Code*)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (*Include Security Classification*)
An Evaluation of a Test Scheduling Solution(U)

12. PERSONAL AUTHOR(S)
Timothy James Kelly

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (*Year, Month, Day*)	15. PAGE COUNT
Master's Thesis	FROM 06/92 TO 03/93	93 March 01	111

16. SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
Testing, Test Scheduling, Software Development, Software Reliability, Software Testing, Software Test Scheduling.

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

As recognized in the software engineering process, software testing during development is an aspect that must be improved to accurately predict and reduce probabilities of future software failures. A possible method of improving software reliability is to concentrate on the scheduling of the test process to reduce costs and increase coverage.

Software test scheduling is the process of sequencing the test procedures to manage costs and maximize verification and validation of the system being evaluated. Changing the methodologies of software testing by implementing a scheduling process can affect many issues in software testing. Software testing is an evolutionary process; to be effective, the test scheduling problem and solution must be continuously revisited, revised and permitted to change according to the events as they occur. This implies that the test scheduling solution is dependant upon many factors, including software design model, results of previous test(s), and the time and resources available for further testing. This empirical study takes the testing information from a Published Specification and performs a detailed analysis of a scheduled solution. Based on the results of this work, it has been determined that the work and resources required to design and develop a software test schedule outweigh the resulting benefits.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
[X] UNCLASSIFIED/UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Imothy J. Shimeall	22b. TELEPHONE (*Include Area Code*): (408) 656-2509 / 22c. OFFICE SYMBOL: CS/Sm

D FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

Approved for public release; distribution is unlimited

An Evaluation of a Test Scheduling Solution

by
Timothy James Kelly
LT, USN
BBA, Wichita State University, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1993

ABSTRACT

As recognized in the software engineering process, software testing during development is an aspect that must be improved to accurately predict and reduce probabilities of future software failures. A possible method of improving software reliability is to concentrate on the scheduling of the test process to reduce costs and increase coverage.

Software test scheduling is the process of sequencing the test procedures to manage costs and maximize verification and validation of the system being evaluated. Changing the methodologies of software testing by implementing a scheduling process can affect many issues in software testing. Software testing is an evolutionary process; to be effective, the test scheduling problem and solution must be continuously revisited, revised and permitted to change according to the events as they occur. This implies that the test scheduling solution is dependant upon many factors, including software design model, results of previous test(s), and the time and resources available for further testing. This empirical study takes the testing information from a Published Specification and performs a detailed analysis of a scheduled solution. Based on the results of this work, it has been determined that the work and resources required to design and develop a software test schedule outweigh the resulting benefits.

T
A939
A988
c.1

TABLE OF CONTENTS

I. BACKGROUND AND RELATED RESEARCH 1
A. INTRODUCTION 1
B. BACKGROUND 2
C. SOFTWARE TEST GENERATION 4
1. General 4
a. Positive / Negative Tests 5
b. Functional / Structural Tests 5
2. Testing Methods 6
a. Test Methods Related to Requirements Analysis 6
b. Test Methods Related to Functional Specification (Requirements Definition Phase) 8
c. Test Methods Related to Architectural Design 9
d. Test Methods Related to Implementation 10
e. Test Methods Related to Installation/Evolution 11
D. MODEL OF SOFTWARE TESTING 12
E. PROBLEM DESCRIPTION 14
F. OVERVIEW OF THE THESIS 14

II. SOFTWARE TEST SCHEDULE ANALYSIS TOOL 16
A. TOOL OVERVIEW 16
B. TOOL DESCRIPTION 17
C. TOOL IMPLEMENTATION 17
1. Graph Description 17
a. Node Information 19
b. Arc Information 21
2. Example Graph Implementation 22
D. TOOL SOLUTIONS 23
1. Output Description 24
a. Evaluation Cost 24
b. Initiation Cost 24
c. Test Procedure Cost 25
d. Total Graph Cost 25
2. Output Processing 25
E. DESIGN EVALUATION 26

III. SCHEDULE-TOOL IMPLEMENTATION 27
A. INTRODUCTION 27
B. DESCRIPTION OF THE DATA 27

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

C. ASSUMPTIONS AND PRECONDITIONS 28
1. Assumption Related to Costs 29
2. Assumption Related to Nodes 29
D. GENERATION OF INPUT DATA 29
1. Determination of Node Weights 31
a. Evaluation Cost 32
b. Initiation Cost 32
2. Determination of Edge Weights 33
3. Determination of Test Process Graphical Structure 34
E. TOOL DATA GENERATION AND INTERPRETATION 36
1. Predictive Data Characteristics 36
2. Data Evaluation 38
3. Implication of Data 40
F. CONCLUSION 41

IV. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH 42
A. CONCLUSIONS 42
B. RECOMMENDATIONS FOR FUTURE RESEARCH 44

APPENDIX A: Parser 46
APPENDIX B: Scheduling Tool Program 54
APPENDIX C: Graph Package Specification 71
APPENDIX D: Program Variable Aggregates 76
APPENDIX E: Input Data Sets 80
APPENDIX F: Data Set Graphs 83
APPENDIX G: Scheduling Tool Output Data 88
REFERENCES 102
INITIAL DISTRIBUTION LIST 104

I. BACKGROUND AND RELATED RESEARCH

A. INTRODUCTION

Computer involvement in our daily lives is becoming more common and complex. Computer systems have become accepted by society as necessary for improved quality assurance and safety where the potential for loss of life exists. The impact of these largely ignored systems and their associated influence are generally unnoticed until something goes wrong [BOLC 90]. These complex systems require improved verification and validation through software testing to reduce the probability of failure and prevent losses in productivity or human life.

The process of testing software, is an activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component [IEEE 91]. As recognized in the software engineering process, software testing during development is an aspect that must be improved to accurately predict and reduce probabilities of future software failures. At issue is how to prevent or circumvent system failure and increase reliability through improved software test scheduling and design. A possible method of improving software reliability is to concentrate on the scheduling of the test process to reduce costs and increase coverage.

The purpose of software testing is to eliminate faults. Therefore, the only successful test is one that finds a software fault [MYER 79]. Software test scheduling is the process of sequencing the test procedures to manage costs and maximize verification and validation of the system being evaluated. Changing the methodologies of software testing by implementing a scheduling process can affect many issues in software testing. Specifically, once software tests are to be sequenced, how does one solution impact the selection of the next test or series of tests; at what point does a successful test influence the selection of the follow-on or remaining tests to completion of product testing? These issues are important to continued validity of the test schedule during the testing process since once scheduling of tests is initiated, continued updating and modification of the test schedule is necessary. Software testing is an evolutionary process; to be effective, the test scheduling problem and solution must be continuously revisited, revised and permitted to change according to the events as they occur. This implies that the test scheduling solution is dependant upon many factors, including software design model, results of previous test(s), and the time and resources available for further testing. These factors will be addressed as the process of test scheduling is studied.

B. BACKGROUND

There are few existing references to software test scheduling. Previous work has concentrated on the automation of the quality control activities in software development. In automatic software test generation, the functional specification identifies the areas that

must be validated. This allows the development phase and test preparation to be performed concurrently [CAMU 90].

The cost of verification and validation activities can be high, meeting and perhaps exceeding 50% of the overall expenses of the project [CAMU 90]. This places increased pressure on the software test team to identify and locate the maximum number of system errors and faults during the testing phase. The test teams are finding that they must do more with less due to these increasing pressures. Due to the changes taking place in the test phase, current and historical information is more valuable than ever. Information and data must be gathered and retained so as much information as possible is available for improved referencing in test scheduling.

Despite the fact that testing during the development phase is recognized as a critical aspect of the software engineering process, resources remain limited in the testing phase. The resources, including personnel, software tools and computing time and hardware must be shared throughout the development process. Thus testing must become an exact discipline to prevent management from considering it a necessary evil to product development. By scheduling the test process, testers may be able to improve effectiveness through increased efficiency and as a result, reduce verification and validation costs.

C. SOFTWARE TEST GENERATION

1. General

There continues to be an extensive amount of research devoted to the study of software testing. Software test methods are designed according to the current stage in the software life cycle; test methods or processes include specification, design and code reviews, walkthroughs, inspections, etc., to full product acceptance testing. Testing is an unavoidable function of the development process; software testing is the only way to demonstrate that the program performs as designed and specified.

The concept of scheduling is based on the premise that by isolating behaviors, common procedures and functions we can improve verification. To provide the best coverage, values are represented as a class (behavior, shared property or process) and it is assumed that the routine will run correctly for other values in the class, when the related test has been correctly executed.

In this implementation, a test is composed of:

- a set of data to exercise a program.
- a set of expected results from its execution.
- a mechanism (programs, rules given to humans, etc.) to carry the system under test in a predefined state, to submit input data, and to collect output results.
- a mechanism (programs, rules given to humans, etc.) to compare actual results with expected ones.
- a description of test scope, that is, the list of 'elements' (behavior of program physical entities of 'elements' such as variables or control paths, etc.) on which the test has focused [CAMU 90].

Based on these definitions, tests can be classified as either positive or negative, or functional or structural tests.

a. Positive / Negative Tests

A positive test checks that a program works correctly when correctly used (i.e., when the input set of data are provided correctly and in the proper format). These tests verify that the program executes to the set of expected results. The scope of a positive test is to verify the program performs correctly when given correct inputs.

A negative test checks that a program works correctly when misused, i.e., that incorrect inputs (values or formats) are recognized, discarded, and signaled and no crashes or damage occur [CAMU 90]. Negative tests verify that the program recognizes an incorrect set of data and performs as designed. The scope of negative testing is to ensure incorrect conditions generated outside of the tested environment are correctly handled.

b. Functional / Structural Tests

In functional program testing, the design of a program is viewed as an abstract description of its design and requirements specifications. Function and data abstractions are used as guides to identify the abstract functions of a program and to generate the functional test data [DEMI 87]. A functional test checks that the item functions properly when provided to the user.

Structural testing refers to the generation of test data to evaluate each part of the structure of the software. The input set of data is designed to manipulate the program from a structural perspective; a structural test exercises physical parts of the program. A program is working correctly when executing a specific control path.

2. Testing Methods

> It has been observed that the later an error has been detected, the more expensive it is to correct. This observation encourages testing early in the development of software [DEMI 87].

In order to increase validation and verification, testing begins in requirements analysis and continues throughout the software systems operational utility. The methods used in evaluation are implemented and evaluated differently with different goals. The ability to apply the scheduling methodology to any of the development testing strategies is important to ensuring complete applicability of the test scheduling process. The remainder of this section concentrates on the test methods in the development process, their applicability to our definition of a test and how scheduling these processes may influence the result.

a. Test Methods Related to Requirements Analysis

Requirements documents have historically been the most poorly "tested" work products in the development cycle [HETZ 88]. This seems contradictory to the premise that the requirements documents are the foundation for the development of the software product in the first place. Traditionally, the requirements are analyzed using a checklist of correctness conditions, including such properties of requirements as consistency, their necessity to achieve the goals of the system, and the feasibility of their implementation with existing resources[DEMI 87].

Testing requirements involves answering two basic questions: Are any of the requirements missing and can any of the requirements be simplified or eliminated [HETZ

88]. In requirements analysis the data to be tested or evaluated are the issues raised in the specifications of the requirements documentation. Each requirement of the software must be considered for their applicability to achieve the software systems' purpose for development. The expected result of evaluating the requirements documents is to determine whether they have specifically meet the definition of the users needs in the product specification, while minimizing resources, overhead, cost, and maximizing efficiency and productivity in the software and its development.

The methods used in requirements vary from checklists designed for structural reviews and walkthroughs to automated static checking and verification tools. Thus, the scope of requirements testing is process specific; the scope of requirements testing is based on the desires of the user. This may be the reason requirements have historically been the most poorly "tested" work products.

When properly tested and evaluated, the requirements documents can have a much greater influence in the developments process. If the requirements documents have been evaluated (tested) and are relied upon in development, the design of final system acceptance tests based on the requirements can be initiated in the requirements analysis phase. These initial acceptance tests will remain tentative until the final specification version is understood and comprehendible, but their usefulness in understanding the required final product behavior can be important the development processes of the follow-on stages.

b. Test Methods Related to Functional Specification (Requirements Definition Phase)

The elements of a software system specification can be analyzed similar to the one used in requirements analysis. Each property specified in the checklist may be checked by a methods similar to those in the requirements analysis, based on the detailed requirements definition and the approved functional software specification.

The data to exercise the process are the requirements specification. The requirements definition phase is expected to determine the exact processes necessary to meet the specifics of the documentation. The mechanisms used to perform this process include formal and informal methods.

Initial logic testing can begin to determine proof of correctness of specified items of interest, critical logical concepts and areas of undeveloped, proposed functionality. These are useful in evaluating software reliances and module pre-requisites. Formal proving or proof of correctness of the software system is extremely difficult and in many instances, impossible. However, the formal proof of correctness of any module can be influential in the design process and can reduce the testing requirements of related software modules; a module formally proven to be correct ensures reliability and reduces the need for it's module testing. Once a module is formally proven the testing emphasis can be shifted to the calling modules and thus reduces the testing requirements.

The purpose of this phase is to derive requirements-based test situations and use them as a test of requirement understanding and validation. In this process, the scope is

not limited to the specified system and software requirements. The requirements definition phase is not limited to preclude concluding prematurely. Upon completion of the testing and evaluation of the functional specification phase, it is anticipated that the modules have been verified to prevent incorrect implementation in the design phases. If the process is scheduled, it may improve module coverage by sequencing the requirements based on pre-requisite processing.

c. Test Methods Related to Architectural Design

The issues related to software reusability are relevant in the design phase. Concepts that have been previously tested, verified or proved can be applied wherever possible. In circumstances in which this is not possible every effort must be made to utilize reusability in the development of new software designs.

The best management tool that can be employed during this phase is the *structured walkthrough*, which seeks to make the design process even more visible and hopefully leads to a more understandable product. The design phase should initiate development of the integration testing topics and issues. The integration testing phase is dependent upon the reasoning and issues used in the design process. These issues are directly related to the product integration and are more easily understood at this development stage than any other. This knowledge influences the development of the test procedures; at this stage the designers have full comprehension of the requirements and what is necessary for specification validation. The data sets and expected results are easily outlined and the generation of the testing mechanisms to carry the system can be developed

in parallel with system design. Architectural design can have a major influence in the test development process due to the level of specification understanding necessary. The designers impact on the testing mechanisms can include the data set selection, set of expected results, the system test process and the description of the test scope. Architectural design is the single most phase having the greatest influence on the test development process.

Scheduling of the issues relevant to architectural design are important due to the descriptive nature of the evaluation results. The design is evaluated during development, and tested as completed sub-units and as whole packages. The module information provides the relevant pre-requisite data for scheduling of implementation and system test design and test processing.

d. Test Methods Related to Implementation

> Traditionally, coding begins after the completion of the design phase and ends at some arbitrary point. In some ways coding does not end until the software is abandoned. The goal of implementation are to provide, justify, and verify a realization for each module [BERZ 91].

Most testing associated with the implementation phase is performed at the programmers level [DEMI 87]. Hence, much of the testing performed at the implementation level is desktop or programmer walkthrough.

Implementation testing is what many people consider the most important phase in the testing process. The data used in the testing is designed to execute the code to reach every solution designed to be met, reach values outside of the specified ranges,

perform an incorrect sequence of operations or fail to determine a correct solution based on the inputs, etc. In addition to verifying how the software processes, the implementation tests are designed to help locate and change any incorrectly implemented procedures, functions or modules.

Additional testing developed for implementation recognizes that this phase is the installation of the system and requires evaluation of performance based on the description of the internal structure of each module specified in the design phase [DEMI 87]. This implies that continued development of the integration testing platform is continuous. The implementation phase can point to problems or difficulties related to the generation of algorithms and data structures. These problems or difficulties must be considered in the continued development of the module, integration and acceptance test cases.

Scheduling in the implementation phase is dependant on the design issues and the progression of the development cycle. During implementation, the schedule of the modules tested is more dependent on the evolution of coding than anything else. However, the implementation phase provides scheduling input information for the system testing process.

e. Test Methods Related to Installation/Evolution

Software changes made after implementation are more error prone than the initial development, and require more attention to quality assurance techniques at every stage. This is why proper documentation of every phase of test planning development and

implementation is emphasized. Testing in the software evolution phase is more easily performed if the original and subsequent change testing documentation is available. In the instances in which information and data is available, the testing process becomes a follow-on phase adapted from previous work, exercises and data. In the event no previous work exists, the testing process must be developed to meet cost verses benefit considerations.

In installation or evolution testing, the data used to exercise the program is designed to specifically exercise a certain process, function or module. While the results of this testing are predetermined, the evaluation of secondary effects within the system must be performed. The testing of an existing software system is much more difficult than developing and implementing the test program in the software development process. The limitations and pressures in place preclude "starting from scratch". Thus, the test team must consider every effect the changes can have on the system. This complex and detailed procedure requires that the testing process be performed in a tighter and more confined arena, on a much smaller scale, while providing the same level of quality assurance as a complete product development test process performed over the development life of the software system.

D. MODEL OF SOFTWARE TESTING

Software testing has been modeled as a sampling or approximation-of-exhaustion problem, where the concern has been to extensively sample from an identified set of portions of the software's internal topology or behavior. We seek to model the generic task

of software testing, preparatory to examining the effect of scheduling strategies on the task of testing software. In general, the task of software testing has two goals:

- to demonstrate the most important behaviors of the software;
- to identify the behaviors of the software that are considered faulty;

Based on theses two goals, given a program, represented as a set of behaviors $B = \{b_0, b_1, b_2, \ldots b_n\}$, and a set of tests $T = \{t_{a,a}, \ldots, t_{z,z}\}$ with:

- A real-valued function, $s(b_0)$, indicating the cost of program test initialization and start-up, resulting in the test starting behavior b_0.
- A real-valued function, $c(b_i, b_j)$, indicating the cost of $t_{i,j}$, the transition from the i^{th} observable behavior to the j^{th} observable behavior (note, this includes instrumentation costs, execution costs and re-initialization or setup costs);
- A real-valued function, $c(b_j)$, indicating the cost of observing b_j (note, this includes the initiation cost when applicable, required resources, instrumenting, collecting results, result comparison, judging results, and the cost of evaluating the test oracle on b_j);
- A boolean-valued function $i(b_j)$, indicating if b_j is an behavior requiring observation and evaluation as stipulated by the test team;

find the total evaluation, initiation, test and transition cost from each combination of observable behaviors and test processes. This model is designed to represent the process of determining test scheduling costs, and is the basis of the scheduling tool development.

E. PROBLEM DESCRIPTION

Testing of any software system is a formidable challenge. The purpose of this study is to isolate the scheduling problem and show how this impacts the testing strategy. The goal is to develop a strategy and an analysis tool in which test scheduling can be analyzed to determine whether a scheduled testing strategy can reduce the cost of software testing. The end result of this study is to determine the influence of scheduling on the test process.

F. OVERVIEW OF THE THESIS

While there has been extensive research in the area of software testing and automatic software test generation, test scheduling has not been considered. There is little research aimed at examining the impact of scheduling on improving test coverage, validation and verification and reducing test expenses. This first chapter has provided an introduction to the concepts of test scheduling. The remainder of this thesis will deal with the design of a scheduling tool, generation of the input data set for analysis and measurement and analysis of the scheduling data results.

Chapter II is a detailed description of the scheduling tool design and implementation. It includes the description of the input format of the scheduling tool as well as information on the operations performed on the tool input data set.

Chapter III is a description of the data set to the scheduling tool and a presentation of the results. This includes a detailed description, discussion and evaluation of the expected and actual results. All observations made in the development of the scheduling tool, input

test data set and analysis data will be presented. These observations are presented with regard to their effect on test scheduling and the software test process.

The focus of Chapter IV is the effect of the conclusions drawn in Chapter III and recommendations developed based upon the findings of this effort. Included is a discussion of recommendations for future research: to improve the software testing process.

II. SOFTWARE TEST SCHEDULE ANALYSIS TOOL

The process of scheduling tests and the development of test data begins in the system analysis and design process. What is tested and the approach used to ensure adequate coverage is determined using a combination of heuristics, historical information and the test teams' experience.

This chapter describes the design and implementation of the test scheduling tool with an emphasis on the test development concepts related to scheduling. The purpose of the test scheduling tool is assist in the evaluation of completing an in-depth schedule or ordering of the software tests throughout development, including the test and analysis phase.

A. TOOL OVERVIEW

The scheduling tool is developed using AYACC [TABA 88], [NGUY 88] to parse the input data (See Appendix A), and build a graph representation using a reusable software graph package [BOOCH 87] (See Appendix B). Most important to the scheduling tool is the representation of the test data and information as elements of a directed acyclic graph. Once the directed acyclic graph is created and initialized with the input data, separate software is used to perform graph transitions to generate a solution set.

B. TOOL DESCRIPTION

In the development of the scheduling tool, several definitions and descriptions must first be resolved. The program and its tests are represented as a collection of observable behaviors and data evaluation procedures called test procedures. A graph model is used to represent a collection of test procedures as a testing process. The test process is the set of all tests designed to verify or validate a module or data set of the software system. The collection of all test procedures is the software test plan. The directed acyclic graph is a representation of observable behaviors as nodes and test procedures as arcs to delineate the possible test schedule paths. The graph data structure permits many combinations of test procedures to be represented simultaneously.

C. TOOL IMPLEMENTATION

The scheduling tool requires the directed acyclic graph to maintain the testing behaviors and test pre-requisites in a logical and easily understood structure. (See Figure 2.1). The graph data structure was designed using reusable software (See Appendix C) that permits easy modification, traversal and information access of the graphical elements. The following section describes in detail the tool implementations and data representation.

1. Graph Description

The nodes in the graph are a representation of observable behaviors. Each arc is a test procedure signifying the transition of data from one observable behavior to one or more others. All behaviors and tests are uniquely represented and identifiable. Each

observable behavior has a unique identification number, evaluation cost, initiation cost and importance indicator. The test procedures are identified by an integer identification number

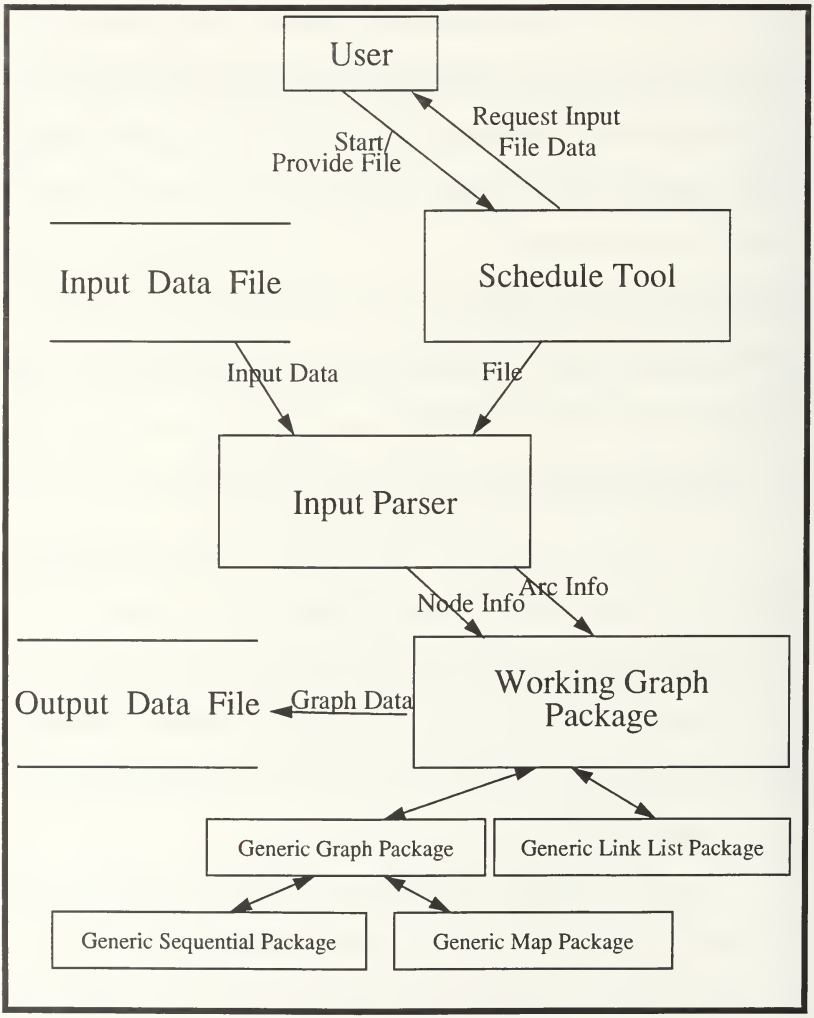


Figure 2.1 Schedule Tool Flow Diagram

and the list of the nodes connected, annotated by the cost of each transition. The node and arc structures are described as follows:

a. Node Information

The node is a representation of the requirement behavior. The following are the characteristics of the node data structure.

(1) Node Identification: The identification number is assigned by the test team. It is an indicator used to sequence the nodes and provide a reference label.

(2) Node Cost: The cost represents the reduction against the total allowed to perform the test procedure. The cost is a value that can give the user a measurement of progress against the total units allocated.

The node cost is composed of two element values: Evaluation and Initiation cost. The inputs to each of the values is different and their summation is not a single representative value useful in this implementation; the evaluation and initiation costs have different purposes and utilizations. The cost variable is designed for use by multiple test procedures. Once the costs have been accrued by the test procedure there is no associated cost to remain in any state, stationary or otherwise.

(a) Evaluation Cost: In every instance of an observable behavior there is an evaluation cost. The node evaluation cost is the representation of the resources required to evaluate the observable behavior. The evaluation cost is represented as a single comprehensive value; the evaluation cost is a measure of the work and resources required to verify the process was completed, the results meet the post-condition requirements and

determine if the test case is (was) correctly implemented. Assumptions made in the design of the graph representation relate to initialization and completion of evaluation processes. Starting cases are assumed to have a built-in allowance for any initialization or start-up requirement costs. In all ending test cases, the comprehensive procedure evaluation cost is subsumed by the node evaluation cost.

Each case of a test procedure has different associated costs; each of the costs is an estimate of the cost of collecting results to analyze the data and the cost of judging these results. The cost of collecting results is proportional to the size and frequency of the views of the observable behavior or condition. The cost of judging the data is directly related to the number of data comparisons necessary for evaluation.

(b) Initiation Cost: In a directed acyclic graph, any node can be the starting node. Because of this, each node has an initiation cost that is a measure of the resources required to begin a test procedure from that state.This is the cost associated with starting in a state other than an identified start state having no pre-requisite requirements.

(3) Importance Factor: In every sequencing system, certain processes have higher priorities for completion. The test cases having higher priorities in the test procedures are so indicated by a boolean importance indicator. The indicator is initialized to false in the input data set. After initialization, the graph is traversed and a graph of only test cases having a positive importance indicator is created called the importance graph. The importance graph is a reference graph of test cases requiring completion and evaluation. This graph is the launch point for graph expansion and data generation.

b. Arc Information

An arc in the graph is a representation of a test procedure. The test procedure is the activity in which a system or component is executed under specified conditions. Each arc represents the execution activity and the transition from one observable behavior to another.

(1) Arc Identification: Similar to the node identification element, each arc has a unique identification number. The identification number is a reference integer used to sequence the arcs and provide an address label. Each arc is numbered based on the integer identification numbers of the adjacent source and destination nodes of the test procedure in the input set; the arc identification number is the concatenation of the source and destination node identification numbers. In addition to the arc number the arcs have a identification sub-number used as a link to the test procedure reference number. The test procedure number is the test identification number given as an input variable. The arcs have two identifiers to allow multiple traversals from one test case to another under differing circumstances. This allows for zero or more arcs between any two nodes.

(2) Arc Cost: The transition cost from one test case to another is the arc weight or cost. The arc cost is a representation of the functionality relative to a transition from one test case to another. This is the representative cost of execution of the test procedure. The change of state is the change in the data being processed in the test plan. The test procedure cost is an estimate based on the complexity of the issues and data. The arc cost is also a reduction against the resources available to complete the test plan.

2. Example Graph Implementation

The example graph below demonstrates the node and arc adjacencies. (See Figure 2.2.) In this example there are six nodes (test cases) and seven arcs (test procedures). In addition, there are four test processes as follows: 132, 135, 146, 25. These process paths can be reviewed while traversing from unit to unit. The two arcs between nodes 1 and 3 are from different test processes (paths). Each has the same arc identification, but different test identifications to allow accurate cost generation.

The solid circled nodes indicate those states in which it is important for the testing phase to transition through (important nodes). These states could be considered failure points in the test phase; if it is not possible to reach the important units (minimal graph),

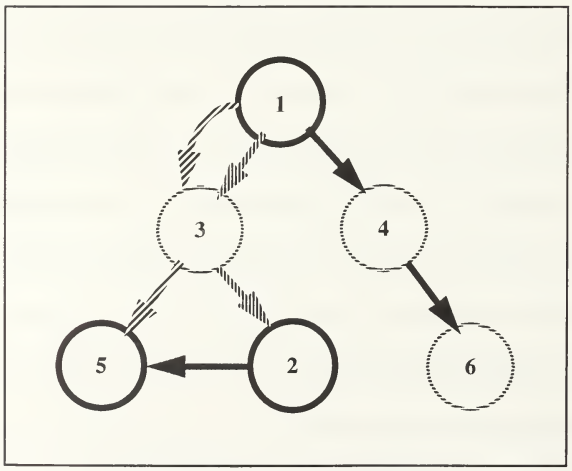


Figure 2.2 Sample Tool Graph

the test phase cannot be completed. In this case either the limit must be modified or the costs must be adjusted.

Based on these descriptions it is apparent that the total cost of any one test procedure is the summation of the costs of the nodes and the associated arcs traversed. Thus the total cost of the test plan is the compilation and summation of all test procedure costs.

D. TOOL SOLUTIONS

The purpose of the scheduling tool is to model the task of software testing, to assist in examining the effect of scheduling strategies and their related costs. The processing by which solutions are generated is based upon a recreation of the graph (working graph) representing the testing process (test graph). The recreation begins from an initial disconnected graph consisting of only nodes indicated as being important. This initial graph (importance graph) is the baseline graph and is evaluated for a cost value. The baseline is the cost to evaluate the behaviors, not including the tests to reach them. The working graph and the test graph are then compared for arc source and destination set commonalities. Consider each ordered pair of vertices v, w in the working graph. Add the edge $v \rightarrow w$ to the working graph if the edge $v \rightarrow w$ also appears in the test graph. After each addition of an arc, the graph cost is computed and graph information is generated. Additionally, once all arc set comparisons are completed, total graph cost information is generated for the node set. After computation of graph cost data for a node set, while the graphs are not equal, a node from the test graph not in the working graph is copied to the working graph. This is

done for every combination of arcs and nodes in the graph and continues until the working graph is equivalent to the test graph. This process generates every possible combination of arcs and nodes in the test graph and provides the scheduling cost data information for analysis information The following is a detailed description of the tool output.

1. Output Description

The output from the scheduling tool is placed on standard output for immediate evaluation. Each line has a field representing the number of nodes, the number of arcs, the total evaluation costs, the total initiation costs, the total test procedure costs and the total graph cost. The cost figures are as described in the node and arc information sections and are computed as follows:

a. Evaluation Cost

The evaluation cost is the cost of evaluating the individual observable behaviors of the graph. This includes the cost of required resources, instrumenting, collecting the results, all result comparisons, judging results, and the cost of evaluating the test oracle specified in the input data set. It is computed as the summation of all the evaluation cost variables of the nodes in the graph.

b. Initiation Cost

The initiation cost is the cost to evaluate an observable behavior without previously performing the pre-requisite test procedures; the initiation cost is the cost of observing a behavior in a state other than an identified start state having no pre-requisite requirements. It is computed as the summation of the initiation costs associated with all

nodes in the graph observed without performing the required pre-requisite observations. This would occur in a disconnected graph in which some node or nodes is an element of the graph but is not reachable via an arc traversal. The tool cost summation procedure identifies every node in the graph not designated as a destination node of any arc. The identified node initiation costs is the Initiation Cost of the graph.

c. Test Procedure Cost

The test procedure cost is computed by totalling the arc cost in the graph. The graph is traversed to identify those arcs having a source and destination arc in the graph. The identified arc costs are summed to determine the total procedure cost in the graph.

d. Total Graph Cost

The total graph cost is the summation of the evaluation costs, the initiation costs and the test procedure costs in the graph. This value represents the total cost to evaluate the observable behaviors in the graph.

2. Output Processing

The raw data generated from the scheduling tool is processed for evaluation using a statistical analysis program. The program is designed to determine node and arc class data. For each combination of nodes and arcs, evaluation, initiation, procedure and total cost values are processed. The minimum, mean, maximum and standard deviation values for the initialization, test and total costs are calculated for each combination. This combined data is grouped by the number of nodes in the graph and sequenced by the number of arcs. The combined analysis data is the basis for the test scheduling evaluation.

E. DESIGN EVALUATION

The graph representation allows for more flexibility in retaining the concepts of a test phase. This structure provides a simple traversal of the graph to evaluate necessary elements and calculate costs. This allows for specific data generation applicable to the study of software test scheduling.

The graph design is implemented in Chapter III. In the implementation, the node and arc element values are described in detail to complete the representation of the input values in the data structure. In addition to specifying the graph implementation details, the input set is operated upon using the scheduling tool and subsequently evaluated. The results of the evaluation of the scheduling data of the test set are concluded upon in Chapter IV.

III. SCHEDULE-TOOL IMPLEMENTATION

A. INTRODUCTION

This study analyzed the patterns of graph costs of software testing plans. The primary goal of this research was to develop a process to generate all possible schedule solutions in a graphical representation of a testing process and determine what relationships exist; the goal of this study was to develop a strategy and a scheduling analysis tool in which the test scheduling solution can be identified for the purpose of reducing the complexity and burden of software system and product testing.

This chapter describes how the data that was used was reduced to a form useful for analysis. The graphical representations (including node and edges descriptions) and input data set examples are presented. Finally, a schedule analysis solution is discussed.

B. DESCRIPTION OF THE DATA

The data taken for the examination were generated based on a specification of a program called *Conflict* that simulated combat interaction between two armies. The specification required the program to accept global data describing the position, size, and attributes of each army plus a description of the terrain operations were to be conducted on. The specification requires the program to return descriptions of the encounters between armies, plus a final condition of each army [SHIM 87].

Each army is composed of one or more battalions, which are made up of one or more squadrons. The program is given global data concerning information on the initial location and attributes of each location. Encounters between the opposing armies are based on the description of their individual battalions as well as the environmental conditions of weather and terrain. Battalions are able to perform five distinct functions: attrition, restoration, movement, communication and observation [SHIM 87].

The test process modeled for this study was based on the testing requirements designed for the movement operation. The testing requirements required the introduction requirements, functional requirements and other individual aggregate variables for precondition purposes. The data generated for this application was based on a single application. The application test plans were prepared by undergraduate students in fulfillment of course work requirements and as testing research material and data [SHIM 87].

C. ASSUMPTIONS AND PRECONDITIONS

As with any research effort, there were certain assumptions made during the course of the development of the methodology of defining the input data. This study is based on a single application of a testing process. The application test plans were prepared by students in fulfillment of undergraduate course work requirements. Professional testers might have approached the test design process differently; a professional test team may have broken down the functionality differently and to a greater detail. The assumptions made in the

development of the data concern the application of cost parameters, both the direct and indirect applications through the assignment of values to initiation and evaluation cost elements. As will be discussed, these assumptions are important to the overall evaluation of this study.

1. Assumption Related to Costs

The cost used in this application is a function of the variables in the testing procedure. Cost is considered a measurement for evaluation only. There is no assumption of accuracy to some actual measure. Several factors in cost consideration are assumed to be distributed evenly to all operations as overhead: tester training, operator training, node identification, arc identification and test reporting. While these are not trivial costs, they are irrespective of order of test application. The costs used in the calculation of node and arc weights are the schedule dependent costs only. The schedule independent costs are included as overhead, and are assumed to be distributed over the entire test process.

2. Assumption Related to Nodes

It is assumed that each behavior is an identifiable subunit, so initiation and termination of behavior are both included in each node. Normal output (final location and final comparison) and error messages are always assumed to be viewed.

D. GENERATION OF INPUT DATA

In addition to the description of what is represented by each element, a description of the data, based on the test plan for the program *Conflict* is included. The information taken

from the application of the specification *Conflict* was obtained through the evaluation of secondary resources of the test plan. These unpublished resources included a *Test Variant of the Battle Simulation Specification* [SHIM 87], a *Battle Simulation - Test Precis* [SHIM 87] and all applicable test procedures.

The *Test Variant of the Battle Simulation Specification* breaks down the testable requirements delineated in the CONFLICT specification by highlighting the areas of interest to test. The *Battle Simulation - Test Precis* classifies the identifiers used in the *Test Variant of the Battle Simulation Specification.* There are five possible symbols used in the *Battle Simulation - Test Precis*: "A", "I", "T", "NTR" and any combination thereof. An "A" is used to indicate that the identifier is a requirement needing careful analysis of the source code. An "I" indicates that the requirement can be verified from the source code. A "T" indicates that the requirement can be verified by a run-time test. The symbol "NTR" indicates a non-testable requirement. If two symbols were used, separated by a slash, then either class applied, depending on how the test planner decides its appropriateness to meeting the intended purpose of the identifier. The requirements were ordered in the same order as they appeared in the *Test Variant of the Battle Simulation Specification.* An excerpt of the MOVEMENT operation identifiers are illustrated in Figure 3.1.

The descriptions of the data items include an explanation of its composition and the referencing resources. Each data item is made up of several inputs, requiring different computations based on the resource documentation it was generated from. The inputs are a

reflection of the descriptions of the scheduling tool, cross-referenced to the Model of Software Testing described in Chapter I, applied based on the specification.

1. Determination of Node Weights (C_i)

The weight of any node is the cost to evaluate the i^{th} observable behavior, including instrumenting and comparing results. Based on the specification, the movement

Name	Class	Explanation of Test Method (if needed)
WhenMove	T	Run a scenario where movement of all battalions would occur at time t, then verify that such movement did occur
HowMove	I	
ThetaIn	T	Run scenario in which Θ_g is the value of army then verify that it is accepted as input describing the battalion g.
ThetaMeasure	A	Analyze code and verify that Θ_g is measured in radians (check inout scaling and correct scaling in equations involving Θ_g).
VgCalc	T	Run scenario with irregular terrain/ weather. Subsumable.
SquadVCalc	I	Verify that the squadron velocity is the result of two factors: initial squadron velocity $(V0_{g,j})$ and the fraction of initial endurance remaining.
WhenVg	T	Run scenario and observe that at each time t, the velocity of each battalion (V_g) is determined.
Borders	T	Run a scenario in which a battalion is given the opportunity to go outside the borders specified in the terrain array and verify that the battalion does not go outside the borders.
WhenBorders	T	Run a scenario an observe that at each time t, the battalions are located inside the appropriate borders

Figure 3.1 - Test Precis Excerpt

aggregates were identified and categorized (See Appendix D). The test evaluation requirements were reviewed to determine the number of comparisons necessary to evaluate the behavior based on the number of variables in the test procedure. The node cost was calculated by the multiplication of the sum of test cases, executions, revisions and modifications of the test scenarios by the number of variables in the test procedure. The total node cost is a summation of the evaluation and initiation costs.

a. Evaluation Cost (C_x)

The evaluation cost assigned to each node is the summation of the cost of collecting and judging results. The cost of collecting results is a function of the size (frequency) of the views of the variables. The cost for each node was determined based on the number of variable references made in the procedure requiring validation.The cost of judging results is a measure of the number of data to be compared at a specific time *t*, for each time *t*. The cost of judging results was calculated based on the number of variables requiring evaluation multiplied by the number of variable verification times for each test evaluation required by the test procedure.

b. Initiation Cost $(C_{i,x})$

The initiation cost assigned to each node is the measure of what it would cost to start from the node prior to completing the required pre-requisites at any time *t*. This includes the cost of instrumenting and the cost of executing enough to reach node *x*. The cost of instrumenting is determined based on the number of test data sets, the number of

views and the number of views needed to access needed pre-requisites. The cost of executing enough to reach node x is a function of the number of data sets and the total number of operator actions required by each pre-requisite.

The representation of the data for input was made based on the costs calculated from these descriptions. The input data set included a line for each observable behavior in the test plan. The input line is ordered by the sequential assignment of behavior identification numbers, each having the specified importance factor, the initiation cost and finally, the evaluation cost. The behavior numbering was based on the pre-requisites and testing requirements determined in the behavior cost generation. See Figure 3.2 for an example of observable behavior input line.

2. Determination of Edge Weights $(C_{i,j})$

The weight of any arc is the cost to transition from the i^{th} observable behavior to the j^{th} observable behavior. The edge weight is the summation of the cost of adding instrumentation needed by i but not j, and the cost of executing enough to reach j from i.

```
suspicious 0   0.00   0.00
suspicious 1  26.00   9.00
suspicious 2  36.00  32.00
important  3   4.00  17.00
important  4 768.00  16.00
suspicious 5  42.00  26.00
important  6  30.00  13.00
```

Figure 3.2 - Example Node Description Input

The cost of additional instrumentation needed by *i* but not *j* is the number of views needed to access *i* (the number of views of *i)* not required by *j*. This was found by comparing the required variables of the previous observable behavior to the current behavior requirements and calculating the difference. This difference was the instrumentation needed by *i* but not *j*. The cost of executing enough to reach *j* from *i* is the summation of the number of new data sets in *j* not in *i*, the number of modifications of old data sets and the additional operator actions required by *j*. This information was found in the test procedures as the number of data sets to be generated by the procedure for each case, the number of variables modifications for each case and the number of operator actions necessary for each test case in the test procedure.

3. Determination of Test Process Graphical Structure

The graphical structure of the testing process is based on the requirements of the test procedures. Each test procedure of the Movement Operation was reviewed to assess the necessary pre-requisites. The pre-requisite of any test procedure is a function of the behavior being observed, the modification procedures operating on the variables and the desired conditions of the observable behavior. In the *Conflict* Specification, behavior requirements are specified to include previous operations on variables, current procedures and descriptions of the behavior of the program. In addition to the Specification, the test procedures identified pre- and post-processing requirements. These were the inputs to the design of the graphical structure. Based on all the information, the test cases were represented in the input set as a logical string consisting of the source behavior

identification number, a test cost and the destination behavior identification number (See Figure 3.3).

The application of these data generation specifications resulted in the creation of four input data sets (See Appendix E). The four data sets were created from the same input variables with modifications to the importance factors. In the first data set, the importance factor of every leaf node of the graph is set to important to simulate the requirement of observing the lowest levels of the requirements. The second data set placed importance on the pre-requisite nodes. Theses nodes are characterized as having one or more post conditional behaviors. In the third data set, the sub-tree root nodes are classified as the important behaviors for observation. Lastly, the fourth input data set classifies the sub-tree nodes as important for observation with a modification to the node set. In addition to the change of importance factors in the fourth data set, the set was modified to include an **or** conditional path. This path was added to enable us to review the **and** and **or** pre-requisite conditional paths.

```
TEST 01   0  9.00  1
TEST 12   1 55.00  2
TEST 214  2  5.00 14
TEST 13   1  8.00  3
TEST 16   1  7.00  6
TEST 09   0 33.00  9
TEST 910  9 21.00 10
```

Figure 3.3 - Example Test Input

The input sets provided the pre-requisite information necessary to generate the graphical representations (See Appendix F) and the sequencing of test information for processing.

E. TOOL DATA GENERATION AND INTERPRETATION

1. Predictive Data Characteristics

Based on the design of the data structure and the cost assignment criteria there are predictive characteristics of the output data. Specifically, there are four (4) types of cost calculations associated with the graph cost summation. Each of these are illustrated in Figure 3.4. A description of each case is as follows:

(a) Behaviors A and B are unrelated. The cost calculation is:

$A + (B = A + \varepsilon) = 2A + \varepsilon$, for some cost ε.

(b) Behavior B is a functional follow-on to A. The cost calculation is:

$A + (A + \alpha) = 2A + \alpha$, for some test cost α.

(c) Behaviors A **and** B are both functional follow-on to C. The cost calculation is: $C + A + B = C + (C + \alpha) + (C + \beta) = 3C + \alpha + \beta$, for some test costs α and β.

(d) Behavior C is a functional follow-on to either A **or** B. The cost calculation is: $\min((A + \alpha),(B + \beta))$, for some test costs α and β.

An additional case used in test pre-requisite scheduling is the condition in which behavior C is a functional follow-on to A **and** B (not shown in Figure 3.4). The four illustrated cost calculations are consistent with the anticipated testing behaviors and the

data structure of the tool. The nature of the testing problem reveals that these are the only combinations used in scheduling. The combinations are due to the sequential nature of the testing process in which either an ordering of behaviors is followed or the pre-requisites of a behavior require the preprocessing of specific tests. In those cases where it is desired to observe a behavior prior to completing the necessary pre-requisites, the tests and costs of preparation are assumed by the behavior itself in the initiation cost variable. The graph structure allows for many possible path combinations, and as previously discussed allows for any node or observable behavior to be a starting state. Because of this, the initiation cost is designed such that a behavior requiring several levels of pre-requisite tests can build up to the necessary state of preparedness.

The inherent building nature of testing allows for predictive patterns to develop. We can expect a binomial distribution of the output data due to the evaluation-initiation cost trade-off: When all completed precondition tests for some observable behavior are completed, there is no initiation cost. In cases where the preconditions have not been completed prior to observing a behavior, the initiation cost is applied as the estimated pre-

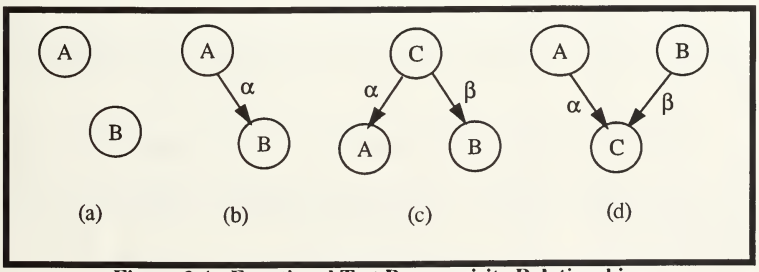


Figure 3.4 - Functional Test Pre-requisite Relationships

requisite testing and evaluation costs. There are many possible variations we may see in the distribution of the data. The two most likely are narrow and broad distributions representing the effects of the behavior of the model evaluation-initiation cost trade-offs. A narrow distribution would represent a situation in which the difference between the test schedule of all the nodes and the test schedule of only the important nodes is small enough such that scheduling may be of no benefit to improving the testing process. If however, the difference in the graph costs is widely distributed (broad distribution) over the cost of performing the test schedule, the test scheduling solution may reduce the overall cost and improve the test coverage of the software testing phase or process.

2. Data Evaluation

Data generated based on the four input sets is presented in total in Appendix G. A review of the output data initially indicate the following characteristics:

- All sets of node/arcs combination sets appear to have a similar shape or distribution.
- The data has a binomial distribution (Narrow Distribution).
- The mean cost in each data section appears to be close to the min/max midpoint in every cost combination.
- The variation about the mean is consistently small relative to the overall total cost.

For analysis purposes we will concentrate on one segment of data set four. Data set four was modified to include a unique test path to evaluate the effects of having an **or** conditional requirement. This was added to verify scheduling consistency and to enable us

to determine any unique or different schedule cost patterns. The output data set used for detailed evaluation is listed as the last section of data output four in Appendix I and is repeated in Table 3.1.

The output data is consistent with the data evaluation characteristics discussed above. The data demonstrates that the variation in the total cost minimum and maximum is small as a percentage of the total cost. For example, the difference between the maximum and minimum costs for a graph having 20 nodes and 11 arcs (line 12 of Table 3.1) is 270. As a percentage, 270 units is 9.2% of the maximum total cost. While not insignificant, this

Table 3.1 - Sample Output Data

		Initiation Cost				Test Cost				Total Cost			
Num	n/a	min	max	mean	StD	min	max	mean	StD	min	max	mean	StD
1	20/0	596.0	596.0	596.0	0.0	0.0	0.0	0.0	0.0	2945.0	2945.0	2945.0	0.0
22	20/1	520.0	589.0	565.7	18.0	3.0	55.0	15.5	14.0	2911.0	2968.0	2930.2	15.0
231	20/2	456.0	584.0	535.7	25.0	6.0	99.0	31.0	19.3	2877.0	2968.0	2915.7	20.7
1540	20/3	401.0	577.0	506.0	30.1	11.0	132.0	46.5	23.1	2843.0	2968.0	2901.5	24.7
7315	20/4	359.0	568.0	476.6	34.1	16.0	165.0	62.0	26.0	2811.0	2968.0	2887.6	27.7
26334	20/5	318.0	560.0	447.5	37.3	23.0	195.0	77.5	28.2	2779.0	2967.0	2874.0	30.1
74613	20/6	277.0	551.0	418.7	40.0	30.0	216.0	93.0	30.0	2750.0	2967.0	2860.7	32.0
170544	20/7	240.0	538.0	390.2	42.2	37.0	230.0	108.5	31.3	2723.0	2964.0	2847.7	33.5
319770	20/8	207.0	522.0	362.1	44.0	45.0	243.0	124.0	32.4	2704.0	2961.0	2835.1	34.7
497420	20/9	175.0	505.0	334.2	45.4	53.0	255.0	139.5	33.1	2692.0	2958.0	2822.7	35.6
646646	20/10	143.0	480.0	306.6	46.4	61.0	264.0	155.0	33.5	2683.0	2952.0	2810.6	36.2
705432	20/11	117.0	463.0	279.4	47.0	69.0	272.0	170.5	33.6	2674.0	2944.0	2798.9	36.5
646646	20/12	91.0	437.0	252.5	47.3	77.0	280.0	186.0	33.5	2665.0	2935.0	2787.5	36.5
497420	20/13	74.0	411.0	225.8	47.2	86.0	288.0	201.5	33.1	2657.0	2923.0	2776.3	36.2
319770	20/14	57.0	379.0	200.0	46.6	98.0	296.0	217.0	32.4	2651.0	2909.0	2765.5	35.7
170544	20/15	41.0	347.0	173.5	45.6	111.0	304.0	232.5	31.3	2648.0	2897.0	2755.0	34.8
74613	20/16	28.0	314.0	147.8	44.1	125.0	311.0	248.0	30.0	2648.0	2878.0	2744.8	33.5
26334	20/17	16.0	277.0	122.4	41.9	146.0	318.0	263.5	28.2	2648.0	2851.0	2734.9	31.8
7315	20/18	7.0	236.0	97.3	39.0	176.0	325.0	279.0	26.0	2648.0	2822.0	2725.3	29.5
1540	20/19	0.0	195.0	72.5	35.1	209.0	330.0	294.5	23.1	2651.0	2790.0	2716.0	26.5
231	20/20	0.0	140.0	48.0	29.7	242.0	335.0	310.0	19.3	2659.0	2758.0	2707.0	22.4
22	20/21	10.0	76.0	23.9	21.8	286.0	338.0	325.5	14.0	2667.0	2724.0	2698.4	16.4
1	20/22	0.0	0.0	0.0	0.0	341.0	341.0	341.0	0.0	2690.0	2690.0	2690.0	0.0

leads us to believe that as a percentage of the total cost, the difference in any test schedule solution is insufficient for an extensive search for the optimal schedule; with a small variation from the smallest to the largest cost, it leads us to believe that the expense and resources needed to develop the test schedule would exceed any realized scheduled improvements and savings.

In addition to the total cost variation, the data implies that any of the test schedule combinations is suitable under a small deviation from the mean; a test schedule cost for each combination of nodes and arcs, tests and observable behaviors is generally within a small range from the mean of the total cost of the graph having all behaviors and tests included. The total cost information on line 12 of Table 3.1 shows that the mean of the graphs (2798.9) is 10 percentage points below the average cost of 2809 [(max+min)/2]). Hence, the distribution of the graph costs is narrow with a majority of all graphs within close distance from the mean.

3. Implication of Data

As discussed in the previous section, the data indicate that there is little variation in the costs of any combination of test schedules. Based on the nature of the testing process, we could expect the trade-offs' in the initiation/evaluation. The testing process drives the binomial behavior due to the cost considerations and pre-requisite conditions of each observable behavior. In every instance, either the pre-requisites have been completed, thus assuming the cost consideration in the previous evaluations and test, or the observing behavior must perform the pre-requisite tests and observations and assume the cost of set-

up in the initiation cost variable. Because of this, the difference in cost between the most optimal schedule and the unscheduled process does not appear large enough to warrant test scheduling. The necessary details required to develop a test schedule solution do not provide an adequate return to a software tester.

F. CONCLUSION

This chapter has presented the implementation details and results of this study. It has also suggested that the results of this study demonstrate that the difference between an optimal testing solution and a loosely based sequence of tests is small enough relative to the total scheduling cost not to warrant test scheduling. The last chapter summarizes the findings of this thesis and discusses how these findings support or contrast performing a scheduling analysis process to the software test plan. In addition to a review of current findings, a discussion of possible follow-on study and directions for further research is presented.

IV. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

The purpose of this thesis is to develop and evaluate a test schedule solution. In addition to increasing the efficiency of testing, it was proposed that the process of scheduling the test set could provide several advantages during software development. In the course of analyzing the solutions, it was determined that the time and expense involved in developing the input set outweigh any savings in improved scheduling of the test procedures. While the results of this work do not appear promising, the experimental population was small, narrowly focused and based on a single testing implementation.

A. CONCLUSIONS

This thesis has been the first examination of a software test scheduling solution. The observations and findings presented in the previous chapters provide a basic knowledge about the process and a limited evaluation of its results. As discussed in Chapter III, there are limitations to the implementation usefulness of the solution. Despite these limitations, this study has provided a basis from which some general conclusions may be made and several areas for future research.

As expected, the output data was binomially distributed. This is apparent based on examination of the Standard Deviation about the mean for every output node set. The binomial distribution results from the model evaluation-initiation cost trade-offs. This

distribution of the cost data by the number of arcs is narrow for each node set in each input data set. As was previously discussed, a narrow distribution of cost data represent a situation in which the difference between the test schedule of all the nodes and the test schedule of only the important nodes is small enough such that scheduling may be of no benefit to improving the testing process. In every node set the minimum cost to evaluate the important behaviors is within a small percentage (<15%) of the maximum cost. Additionally, despite the fact that the total cost of performing all the test procedures is less than the total cost to perform no test procedures, the difference is small enough to generalize that any combination of arc may be optimal. Based on the distribution, and the data comparisons made in Chapter III, an analysis of the functionality of the model and the performance data, a test schedule is not beneficial to the testing process.

While this research has identified difficulties in implementing the testing process graphically, further development of this process may enhance an understanding of the test scheduling process in the software testing environment. The solutions generated based on the input set used in this study do not indicate that scheduling will improve coverage through increased efficiency and a reduction in costs. However, even if this solution provided a noticeable optimal schedule cost it would not be sufficient to imply without limitations that scheduling improves the testing process. This study demonstrates that testing is a very complex process requiring significant resources and efforts of the software development process.

B. RECOMMENDATIONS FOR FUTURE RESEARCH

The results of this study indicate that there are several areas of research suitable for further research. The goal of this thesis has been to develop information useful in the software development process. To this end, a final analysis of test scheduling cannot be made prior to evaluating variations in the testing pre-requisites, assignment of initiation requirements and the cost categories.

The nature of testing has limitations on the assignment of preconditions to a test or an observable behavior. While this cannot be changed, an evaluation of pre- and post-conditions can be made to determine if a sequential ordering is necessary. There are instances in the testing process in which pre-requisites must be completed. If required behaviors and tests are performed as a logical group in stages, this may reduce the set of tests and behaviors requiring sequencing. This is implied by the model, but in practice it may be very difficult to implement.

The variations of evaluation variables may impact the scheduling solutions. A single unit cost was assigned to the variables for each operation. No variation in unit assignment was made based on the complexity of the variable or process. This implies that there exists a one to one correspondence in the cost of performing the tests, observing the behavior, analyzing the results and comparing the results. In actuality this is most likely not correct. A study of differing cost unit assignments may demonstrate a closer relationship between the unit assignments and a scheduling solution.

Consideration must be made regarding the selection of cost data generation. Due to the development process, there are many possibilities in developing the cost basis for data design in test scheduling. An examination of the different development processes and the effects on the data generation may be of interest. Specifically, an analysis of the prototyping and spiral development models could provide insight to improving the test and testing design process.

Software test scheduling does not appear to improve or benefit the testing process. There are several areas in which different applications in the schedule development that may improve the scheduling design and thus increase the benefit to the testing process. Until these new applications are developed, applied and studied, no economical procedure is available. Improvements in detailing requirements, assigning cost functionality and managing pre-requisites may have a practicable effect on a schedule of software tests. Until these improvements are made, test scheduling is not a viable solution to reducing costs and improving test verification, validation and coverage.

APPENDIX A: Parser

driver.a

```
with u_env.parser.input_lex_io.input_lex.text_io.Graph_Works;
use  u_env.parser.text_io;

procedure parse is
  in_file_name : string(1..80);
  last         : natural;
  response     : character;
  completed    : boolean := false;

begin
    text_io.put("Enter input file: ");
    text_io.get_line(in_file_name, last);
    input_lex_io.open_input(in_file_name(1..last));
    input_lex_io.create_output;

    yyparse;

    input_lex_io.close_input;
    input_lex_io.close_output;

    Graph_Works.Create_Graph_Of_Important_Nodes;

    Graph_Works.Generate_All_Graphs_And_Costs;


end parse;
```

input.y

```
-- Declaration Section
%token NODE_TOKEN TEST_TOKEN LIMIT_TOKEN
%token ID_TOKEN SUSPICIOUS_TOKEN IMPORTANT_TOKEN
%token COLOR_TOKEN COST_TOKEN
%token TOK_OPEN TOK_CLOSE
%start input
-- Declarations that will be written to the token package.
{
  subtype key is integer;
  subtype yystype is integer;
  last_id       : INTEGER := 0;
  arc_cost,
  eval_cost,
  init_cost,
  last_cost,
  limit_cost    : FLOAT   := 0.0;
  conversion    : constant FLOAT := 0.01;
  from_vertex,
  to_vertex,
  current_test  : key;
  first_vertex  : boolean := true;
}

%% -- Rules Section

input:  nodesect testsect limitsect;

node : IMPORTANT_TOKEN ID_TOKEN COST_TOKEN {eval_cost := last_cost;}
       COST_TOKEN {init_cost := last_cost;
       build_item(last_id,eval_cost,init_cost,graph_works.important);} |
     SUSPICIOUS_TOKEN ID_TOKEN COST_TOKEN {eval_cost := last_cost;}
       COST_TOKEN {init_cost := last_cost;
       build_item(last_id,eval_cost,init_cost,graph_works.suspicious);};

nodesect  :  node nodesect | node;

test    :  TEST_TOKEN ID_TOKEN
           {set_test_id(last_id,current_test);} idpair;
```

```
testsect  :  test testsect | test;

idpair    :  ID_TOKEN COST_TOKEN
             {if not first_vertex then
               to_vertex := last_id;
               add_arc(from_vertex,arc_cost,to_vertex);
               from_vertex := to_vertex;
               arc_cost := last_cost;
              else
               from_vertex := last_id;
               arc_cost := last_cost;
               last_id := 0; last_cost := 0.0;
               first_vertex := false;
              end if;}
            idpair | ID_TOKEN
             {to_vertex := last_id;
              add_arc(from_vertex,arc_cost,to_vertex);
              last_id := 0;last_cost := 0.0;
              first_vertex := true;};

limitsect :  LIMIT_TOKEN COST_TOKEN
             {limit_cost := last_cost;};

%%   -- User Declarations

with input_tokens,graph_works;
use  input_tokens;

package parser is

 procedure yyparse;

 procedure set_test_id(last_id  : in graph_works.KEY;
                       test_id  : out graph_works.KEY);


 procedure build_item(last_id    : in INTEGER;
                      eval_cost  : in FLOAT;
                      init_cost  : in FLOAT;
                      node_color : in graph_works.IMPORTANCE);

 procedure add_arc(vertex_1       : in out graph_works.key;
                   attribute_cost : in out float;
```

```
            vertex_2      : in out graph_works.key);

end parser;


with input_tokens.input_goto.input_shift_reduce.input_lex.text_io.graph_works;
use  input_tokens.input_goto.input_shift_reduce.input_lex.text_io;

package body parser is

 package key_io is new text_io.integer_io(graph_works.key);
 package natural_io is new text_io.integer_io(natural);
 package float_inout is new float_io(float);
 package importance_io is new enumeration_io(graph_works.importance);

 num_errors : INTEGER := 0;

 procedure set_test_id(last_id : in  graph_works.KEY;
           test_id : out graph_works. KEY) is

  begin

   test_id := last_id;

 end set_test_id;



 procedure yyerror(s: in string := "syntax error") is
  begin
   num_errors := num_errors + 1;

 end yyerror;


 procedure build_item(last_id    : in INTEGER;
              eval_cost  : in FLOAT;
              init_cost  : in FLOAT;
              node_color : in graph_works.IMPORTANCE) is

  temp_item : graph_works.item;

 begin
```

```
  temp_item.id        := last_id;
  temp_item.eval_cost  := eval_cost;
  temp_item.init_cost  := init_cost;
  temp_item.color      := node_color;
  temp_item.covered    := false;
  graph_works.Add_A_Node(TEMP_ITEM);

 end build_item;


 procedure add_arc(vertex_1       : in out graph_works.key;
                   attribute_cost : in out float;
                   vertex_2       : in out graph_works.key) is

   arc_name : graph_works.name;

   function COMPUTE_ARC_ID(vertex_id_1, vertex_id_2 : in graph_works.KEY)
            return graph_works.KEY is

     arc_key  : graph_works.key := 0;

    begin
     arc_key := vertex_id_1 * 10 + vertex_id_2;
     return arc_key;
   end COMPUTE_ARC_ID;

  begin
   arc_name.id := compute_arc_id(vertex_1, vertex_2);
   arc_name.of_test := current_test;
   arc_name.cost := attribute_cost;
   arc_name.transitioned := false;
   graph_works.Add_An_Arc(ARC_NAME, vertex_1, vertex_2);
 end add_arc;


##%procedure_parse


end parser;
```

input_lex.l

```
%START IDENT Z
A               [aA]
B               [bB]
C               [cC]
D               [dD]
E               [eE]
F               [fF]
G               [gG]
H               [hH]
I               [iI]
J               [jJ]
K               [kK]
L               [lL]
M               [mM]
N               [nN]
O               [oO]
P               [pP]
Q               [qQ]
R               [rR]
S               [sS]
T               [tT]
U               [uU]
V               [vV]
W               [wW]
X               [xX]
Y               [yY]
Z               [zZ]
NODE            [nN][oO][dD][eE]
TEST            [tT][eE][sS][tT]
LIMIT           [lL][iI][mM][iI][tT]
IMPORTANT       [iI][mM][pP][oO][rR][tT][aA][nN][tT]
SUSPICIOUS      [sS][uU][sS][pP][iI][cC][iI][oO][uU][sS]
%%
{NODE}          {ENTER(Z); return(NODE_TOKEN);}
{TEST}          {ENTER(Z); return(TEST_TOKEN);}
{LIMIT}         {ENTER(Z); return(LIMIT_TOKEN);}
{IMPORTANT}     {ENTER(Z); return(IMPORTANT_TOKEN);}
{SUSPICIOUS}    {ENTER(Z); return(SUSPICIOUS_TOKEN);}
[0-9]+  {declare
```

```
        i:          integer;
        mytext:  string(1..100) := (others=>ASCII.NUL);
      begin
        copy(yytext,mytext);
        last_id := 0;
        i:=1;
        while ( mytext(i) /= ASCII.NUL ) loop
           last_id := last_id * 10 + character'pos(mytext(i)) - character'pos('0');
                i:= i + 1;
        end loop;
        if i > 1  then -- we lexed something
           return(ID_TOKEN);
         else
            put_line("Cannot find id matched by lex");
            new_line;
            last_id := 0;
            return(ID_TOKEN);
        end if;
        exception
           when others=>
           put_line("Error in lexing ID");
           new_line;
           last_id := 0;
           return(ID_TOKEN);
      end; -- declare
    }

[0-9]+[.][0-9][0-9] {declare
            i,
            int_part,
            dec_part :integer;
            mytext:string(1..100) := (others=>ASCII.NUL);
          begin
            copy(yytext,mytext);
            i := 1;
            int_part := 0;
            dec_part := 0;
            while mytext(i)/= '.' and mytext(i)/=ASCII.NUL loop
              int_part := int_part * 10 + character'pos(mytext(i)) -
                                character'pos('0');
              i := i + 1;
            end loop;
          i := i + 1;   -- skip the '.' (decimal)
```

```
while mytext(i)/=ASCII.NUL loop
  dec_part := dec_part * 10 + character'pos(mytext(i))  -
                character'pos('0');
  i:= i + 1;
 end loop;
      if i > 1  then    -- we lexed something
  last_cost := float(int_part) + (float(dec_part)*conversion);
  return(COST_TOKEN);
 else
       last_cost := 0.0;
     end if;
exception
  when others=>
    new_line;
    put_line("Exception in COST loops");
    raise;
 end;}

%%

with input_tokens, text_io;
use  input_tokens, text_io;

package input_lex is

  function yylex return token;

end input_lex;

package body input_lex is

  procedure copy(s1: in string; s2:in out string) is
  j:integer range s2'range := s2'first;
  begin
   for i in s1'range loop
     s2(j) := s1(i);
     j := j + 1;
    end loop;
  end copy;

##

end input_lex;
```

APPENDIX B: Scheduling Tool Program

```
with Text_IO,Graph_Package,link_list, SEQUENTIAL_IO;
use  Text_IO;

package GRAPH_WORKS is

 type IMPORTANCE is (IMPORTANT, SUSPICIOUS);
 subtype KEY is INTEGER;

  type ITEM is
   record
    ID       : KEY;
    EVAL_COST : FLOAT;
    INIT_COST : FLOAT;
    COLOR     : IMPORTANCE;   -- COLOR is a holdover from a red/blue designation
    COVERED   : BOOLEAN;
   end record;

  type NAME is
   record
    ID          : KEY;
    OF_TEST     : KEY;
    COST        : FLOAT;
    TRANSITIONED : BOOLEAN;
   end record;

  package TEST_GRAPH is new GRAPH_PACKAGE(ITEM,NAME);
  use TEST_GRAPH;

  package ARC_LIST is new LINK_LIST(TEST_GRAPH.ARC);
  use ARC_LIST;

  FINISH_GRAPH,
  IMPORTANT_GRAPH,
  FULL_GRAPH,
  THE_GRAPH       : GRAPH;
  TEMP_VERTEX_1,
  TEMP_VERTEX_2,
  WORKING_VERTEX   : VERTEX;
  TEMP_ARC        : ARC;
```

```
TEMP_ID         : KEY;
GRAPH_COUNT     : NATURAL := 1;
THE_ARC_LIST    : LINK;

procedure Add_A_Node(TEMP_ITEM  : in out ITEM);

procedure Add_An_Arc(ARC_NAME   : in out NAME;
                     VERTEX_ID_1 : in out KEY;
                     VERTEX_ID_2 : in out KEY);

procedure Print_Current_Vertices(Working_graph : in out GRAPH);

procedure Clear_Covered_Variable(Working_graph : in out GRAPH);

procedure Calculate_Graph_Cost(A_Graph    : in out GRAPH);

procedure Vertex_Compare(CHECKING_GRAPH : in out GRAPH;
                         V_ID           : in KEY;
                         MATCHING_VERTEX: out VERTEX;
                         MATCH_FOUND    : out BOOLEAN);

procedure Arc_Compare(CHECKING_GRAPH  : in out GRAPH;
                      VERTEX_1,
                      VERTEX_2        : in out VERTEX;
                      REF_ATTRIBUTE   : in NAME;
                      REF_ARC         : out ARC;
                      ARC_MATCH_FOUND : out BOOLEAN);

procedure Create_Graph_Of_Important_Nodes;

procedure Create_Disconnected_Graph(REFERENCE_GRAPH    :in out GRAPH;
                                    DISCONNECTED_GRAPH : in out GRAPH);

procedure Find_Arc_Points(AN_ARC        : in out ARC;
                          A_GRAPH       : in out GRAPH;
                          ARC_ATTRIBUTE : in out NAME;
                          VERTEX_SOURCE,
                          VERTEX_DEST   : in out VERTEX;
                          POINTS_FOUND  : out BOOLEAN);

procedure Build_Arc_Link_List(WORKING_GRAPH,
                              FULL_GRAPH    : in out GRAPH;
                              ARC_LINK_LIST : in out LINK);
```

```
  procedure Generate_All_Graphs_And_Costs;

end GRAPH_WORKS;
package body GRAPH_WORKS is

  package key_io is new text_io.integer_io(key);
  package natural_io is new text_io.integer_io(natural);
  package float_inout is new float_io(float);
  package importance_io is new enumeration_io(importance);

--************************************************************************
--This procedure Adds a node to the Graph using the GRAPH PACKAGE procedure
--Add.
--************************************************************************
  procedure Add_A_Node(TEMP_ITEM  : in out ITEM) is
    begin
     Add(TEMP_VERTEX_1, TEMP_ITEM, THE_GRAPH);
  end Add_A_Node;


--************************************************************************
--This procedure Adds an Arc to the Graph with the source and destination nodes
--This procedure uses the GRAPH PACKAGE vertex traversal to locate the vertices
-- of the Arc to be added.
--************************************************************************
  procedure Add_An_Arc(ARC_NAME    : in out NAME;
                       VERTEX_ID_1 : in out KEY;
                       VERTEX_ID_2 : in out KEY) is

    procedure VERTEX_SEARCH (THE_VERTEX   : in TEST_GRAPH.VERTEX;
                             CONTINUE     : out BOOLEAN) is
      TEMP_ITEM : ITEM;
     begin
      TEMP_ITEM := Item_Of(THE_VERTEX);
      if TEMP_ITEM.ID = TEMP_ID then
       WORKING_VERTEX := THE_VERTEX;
      end if;
      CONTINUE := TRUE;
    end VERTEX_SEARCH;
    procedure GET_VERTEX is new
                  TEST_GRAPH.Iterate_Vertices(VERTEX_SEARCH);
```

```
 begin    -- begin for Add_An_Arc

  TEMP_ID:= VERTEX_ID_1; GET_VERTEX(THE_GRAPH);
  TEMP_VERTEX_1 := WORKING_VERTEX;
  TEMP_ID:= VERTEX_ID_2; GET_VERTEX(THE_GRAPH);
  TEMP_VERTEX_2 := WORKING_VERTEX;
  TEST_GRAPH.CREATE(TEMP_ARC,ARC_NAME,TEMP_VERTEX_1,
                    TEMP_VERTEX_2,THE_GRAPH);

 end Add_An_Arc;

--************************************************************************
--This procedure generates a Graph of nodes indicating a positive importance factor.
--The procedure uses the GRAPH PACKAGE vertex traversal procedure to evaluate
--each node in the input graph.
--************************************************************************
 procedure Create_Graph_Of_Important_Nodes is

  temp_graph : TEST_GRAPH.GRAPH;

  procedure IMPORTANT_LOOK(THE_VERTEX   : in TEST_GRAPH.VERTEX;
                           CONTINUE     : out BOOLEAN) is
    temp_item : ITEM;
   begin
    temp_item := TEST_GRAPH.Item_Of(the_vertex);
    if temp_item.color = important then
     TEST_GRAPH.Add(TEMP_VERTEX_1, temp_item, temp_graph);
    end if;
    CONTINUE := TRUE;
  end IMPORTANT_LOOK;
  procedure generate_important_graph is new
          TEST_GRAPH.Iterate_Vertices(IMPORTANT_LOOK);

 begin   -- create_graph_of_important_nodes

  TEST_GRAPH.Clear(FINISH_GRAPH);
  TEST_GRAPH.Copy(THE_GRAPH, FULL_GRAPH);
  generate_important_graph(FULL_GRAPH);
  TEST_GRAPH.Copy(TEMP_GRAPH, IMPORTANT_GRAPH);

end Create_Graph_Of_Important_Nodes;
```

```
--*************************************************************************
--This procedrue is designed to compare two vertices of different graphs to determine
--if they are identical. The procedure uses the GRAPH PACKAGE traversal procedure
--to compare each node of the graphs until a match is found.
--*************************************************************************
 procedure VERTEX_COMPARE(
                                CHECKING_GRAPH : in out TEST_GRAPH.GRAPH;
                                V_ID           : in KEY;
                                MATCHING_VERTEX: out TEST_GRAPH.VERTEX;
                                MATCH_FOUND    : out BOOLEAN) is

  procedure ID_SEARCH(THE_VERTEX   : in TEST_GRAPH.VERTEX;
                      CONTINUE     : out BOOLEAN) is
    TEMP_ITEM : ITEM;
   begin
    TEMP_ITEM := TEST_GRAPH.Item_Of(THE_VERTEX);
    if V_ID = TEMP_ITEM.ID then
      MATCH_FOUND := TRUE;
      MATCHING_VERTEX := THE_VERTEX;
      CONTINUE := FALSE;
     else
       MATCH_FOUND := FALSE;
       CONTINUE := TRUE;
   end if;

  end ID_SEARCH;
  procedure VERTEX_MATCH is new TEST_GRAPH.Iterate_Vertices(ID_SEARCH);

 begin
  if TEST_GRAPH.Number_Of_Vertices_In(CHECKING_GRAPH) = 0 then
    MATCH_FOUND := FALSE;
   else
     VERTEX_MATCH(CHECKING_GRAPH);
  end if;
end VERTEX_COMPARE;
```

```
--*************************************************************************
--This procedure performs an arc comparison between two arcs of different graphs.
--This procedure uses the GRAPH PACKAGE arc traversal procedure.
--*************************************************************************
 procedure ARC_COMPARE(CHECKING_GRAPH  : in out TEST_GRAPH.GRAPH;
                       VERTEX_1,
                       VERTEX_2          : in out TEST_GRAPH.VERTEX;
                       REF_ATTRIBUTE     : in NAME;
                       REF_ARC           : out TEST_GRAPH.ARC;
                      ARC_MATCH_FOUND    : out BOOLEAN) is

  procedure ARC_ID_SEARCH(THE_ARC   : in TEST_GRAPH.ARC;
                          CONTINUE  : out BOOLEAN) is

    TEMP_ATTRIBUTE : NAME;

   begin
    TEMP_ATTRIBUTE := TEST_GRAPH.Attribute_Of(THE_ARC);
    if REF_ATTRIBUTE.ID = TEMP_ATTRIBUTE.ID then
     if REF_ATTRIBUTE.OF_TEST = TEMP_ATTRIBUTE.OF_TEST  then
       ARC_MATCH_FOUND := TRUE;
       REF_ARC := THE_ARC;
       CONTINUE := FALSE;
      else
        ARC_MATCH_FOUND := FALSE;
        CONTINUE := TRUE;
     end if;
    else
       ARC_MATCH_FOUND := FALSE;
       CONTINUE := TRUE;
   end if;
 end ARC_ID_SEARCH;
 procedure ARC_MATCH is new
         TEST_GRAPH.Iterate_Arcs(ARC_ID_SEARCH);

begin
 if TEST_GRAPH.Number_Of_Arcs_In(CHECKING_GRAPH) = 0 then
   ARC_MATCH_FOUND := FALSE;
  else
```

```
      ARC_MATCH(CHECKING_GRAPH);
    end if;
  end ARC_COMPARE;

--*************************************************************************
--This procedure outputs the Vertex data of all vertices in the specified graph
--*************************************************************************
  procedure Print_Current_Vertices(Working_graph : in out GRAPH) is

    procedure print_all_vertices(The_Vertex : in TEST_GRAPH.VERTEX;
                                 Continue   : out BOOLEAN) is
      temp_item  : ITEM;
    begin
      temp_item := TEST_GRAPH.Item_Of(The_Vertex);
      KEY_IO.PUT(temp_item.ID. width => 2);
      TEXT_IO.PUT(" ");
     Continue := True;
    end print_all_vertices;

    procedure output_vertex_ids is new
                TEST_GRAPH.Iterate_Vertices(print_all_vertices);

   begin  --Print_Current_Vertices

    output_vertex_ids(Working_Graph);
  end Print_Current_Vertices;

--*************************************************************************
--This procedure resets the coverage variable in the vertices of the graph.
--*************************************************************************
  procedure Clear_Covered_Variable(Working_graph : in out TEST_GRAPH.GRAPH) is

    procedure Travel_Vertices(The_Vertex : in TEST_GRAPH.VERTEX;
                              Continue   : out BOOLEAN) is
      New_Item,
      Temp_Item   : ITEM;
      Temp_Vertex : TEST_GRAPH.VERTEX;

    begin
      Temp_Vertex := The_Vertex;
      Temp_Item := TEST_GRAPH.Item_Of(The_Vertex);
      New_Item.ID       := Temp_Item.ID;
      New_Item.EVAL_COST := Temp_Item.EVAL_COST;
```

```
    New_Item.INIT_COST := Temp_Item.INIT_COST;
    New_Item.COLOR     := Temp_Item.COLOR;
    New_Item.COVERED   := False;
    TEST_GRAPH.SET_ITEM(Temp_Vertex,New_Item);
    Continue := True;
  end travel_vertices;
  procedure Clear_Covered is new
                TEST_GRAPH.Iterate_Vertices(Travel_Vertices);

  begin
   Clear_Covered(Working_Graph);

end Clear_Covered_Variable;

--*********************************************************************
--This procedure calculates the graph costs for data generation.
--*********************************************************************
procedure Calculate_Graph_Cost(A_Graph     : in out TEST_GRAPH.GRAPH) is

   arc_count,
   vertex_number       : NATURAL;
   vertex_eval_cost,
   vertex_init_cost,
   vertex_cost,
   arc_cost,
   graph_cost          : FLOAT := 0.0;
   display_the_arcs,
   display_the_vertices : BOOLEAN := FALSE;

 --*********************************************************************
 -- This procedure traverses the vertices of the graph and calculates the node
  --evaluation, initiation and the total vertex cost.
 --*********************************************************************
 procedure sum_vertices(The_Vertex : in TEST_GRAPH.VERTEX;
                        Continue   : out BOOLEAN) is
    temp_item  : ITEM;

  begin
    temp_item := TEST_GRAPH.Item_Of(The_Vertex);
    if temp_item.covered then
      vertex_eval_cost := vertex_eval_cost + temp_item.eval_cost;
    else
      vertex_eval_cost := vertex_eval_cost + temp_item.eval_cost;
```

```
    vertex_init_cost := vertex_init_cost + temp_item.init_cost;
  end if;
  vertex_cost := vertex_eval_cost + vertex_init_cost;
  Continue := True;
end sum_vertices;
procedure calculate_vertices_cost is new
                TEST_GRAPH.Iterate_Vertices(sum_vertices);

--*********************************************************************
-- This procedure traverses the arcss of the graph and calculates the arc cost.
--*********************************************************************
procedure sum_arcs(The_Arc  : in TEST_GRAPH.Arc;
                   Continue : out BOOLEAN) is

  temp_attribute    : NAME;
  Dest_Vertex       : VERTEX;
  Dest_Item,
  New_Item          : ITEM;
  Vertex_Located    : BOOLEAN;
  Changing_Vertex_ID : KEY;

 begin
  temp_attribute := TEST_GRAPH.Attribute_Of(The_Arc);
  temp_attribute.transitioned := true;

  arc_count := arc_count + 1;
  arc_cost  := arc_cost + temp_attribute.cost;

  Dest_Vertex := Destination_Of(The_Arc);
  Dest_Item := Item_Of(Dest_Vertex);
  New_Item.ID       := Dest_Item.ID;
  New_Item.EVAL_COST := Dest_Item.EVAL_COST;
  New_Item.INIT_COST := Dest_Item.INIT_COST;
  New_Item.COLOR    := Dest_Item.COLOR;
  New_Item.COVERED  := True;
  TEST_GRAPH.SET_ITEM(Dest_Vertex,New_Item);
  if arc_count = TEST_GRAPH.Number_Of_Arcs_In(A_Graph) then
    arc_count := 0;
    Continue := False;
  else
    Continue := True;
  end if;
end sum_arcs;
```

```
    procedure calculate_arcs_cost is new
                TEST_GRAPH.Iterate_Arcs(sum_arcs);

   begin  --Calculate_Graph_Cost
     if TEST_GRAPH.Number_Of_Arcs_In(A_Graph) = 0 then
       TEXT_IO.PUT("n = ");
       Print_Current_Vertices(A_Graph);
       new_line;
     end if;
     arc_count := 0;
     calculate_arcs_cost(A_Graph);
     calculate_vertices_cost(A_Graph);
     Clear_Covered_Variable(A_Graph);
     TEXT_IO.PUT(" ");
     NATURAL_IO.PUT(Graph_Count, width => 6);
     TEXT_IO.PUT(" ");
     NATURAL_IO.PUT(TEST_GRAPH.Number_Of_Vertices_In(A_Graph), width =>
6);
     TEXT_IO.PUT(" ");
     NATURAL_IO.PUT(TEST_GRAPH.Number_Of_Arcs_In(A_Graph), width => 6);
     TEXT_IO.PUT(" ");
     FLOAT_INOUT.PUT(vertex_eval_cost,fore => 8,aft => 2,exp => 0);
     TEXT_IO.PUT(" ");
     FLOAT_INOUT.PUT(vertex_init_cost,fore => 8,aft => 2,exp => 0);
     TEXT_IO.PUT(" ");
     FLOAT_INOUT.PUT(arc_cost,fore => 8,aft => 2,exp => 0);
     TEXT_IO.PUT(" ");
     graph_cost := vertex_eval_cost + vertex_init_cost + arc_cost;
     FLOAT_INOUT.PUT(graph_cost,fore => 8,aft => 2,exp => 0);
     TEXT_IO.PUT_line(" ");

     Graph_Count := Graph_Count + 1;

  end Calculate_Graph_Cost;
```

```
--*************************************************************************
--This procedure generates a disconnected graph of the input nodes based on the values
--of the input data.
--*************************************************************************
 procedure create_disconnected_graph
                    (REFERENCE_GRAPH    :in out TEST_GRAPH.GRAPH;
                     DISCONNECTED_GRAPH : in out TEST_GRAPH.GRAPH) is
  --*************************************************************************
  --This procedure traverses a REFERENCE GRAPH and adds each node(vertex) to
  -- a working DISCONNECTED GRAPH.
  --*************************************************************************
  procedure ADD_ALL_VERTICES(THE_VERTEX   : in TEST_GRAPH.VERTEX;
                                          CONTINUE     : out BOOLEAN) is
    TEMP_ITEM   : ITEM;
    TEMP_VERTEX : TEST_GRAPH.VERTEX;

    begin
     TEMP_ITEM := TEST_GRAPH.Item_Of(THE_VERTEX);
     TEST_GRAPH.Add(TEMP_VERTEX, TEMP_ITEM,DISCONNECTED_GRAPH);
     CONTINUE := TRUE;
  end ADD_ALL_VERTICES;
  procedure generate_disconnected_graph is new
              TEST_GRAPH.Iterate_Vertices(ADD_ALL_VERTICES);


   begin   -- create_disconnected_graph

    TEST_GRAPH.clear(DISCONNECTED_GRAPH);
    GENERATE_DISCONNECTED_GRAPH(REFERENCE_GRAPH);
 end create_disconnected_graph;

--*************************************************************************
--This procedure searches the input graph to determine the SOURCE and DESTINATION
--vertices og the graph.
--*************************************************************************
 procedure FIND_ARC_POINTS(AN_ARC        : in out TEST_GRAPH.ARC;
                           A_GRAPH       : in out TEST_GRAPH.GRAPH;
                           ARC_ATTRIBUTE : in out NAME;
                           VERTEX_SOURCE,
```

```
                                    VERTEX_DEST    : in out TEST_GRAPH.VERTEX;
                                    POINTS_FOUND   : out BOOLEAN) is

      SOURCE_ITEM,
      DEST_ITEM      : ITEM;
      SOURCE_FOUND,
      DEST_FOUND     : BOOLEAN := FALSE;

    begin
      ARC_ATTRIBUTE :=  TEST_GRAPH.Attribute_of(AN_ARC);
      VERTEX_SOURCE :=  TEST_GRAPH.Source_Of(AN_ARC);
      VERTEX_DEST   :=  TEST_GRAPH.Destination_Of(AN_ARC);
      SOURCE_ITEM   :=  TEST_GRAPH.Item_Of(VERTEX_SOURCE);
      DEST_ITEM     :=  TEST_GRAPH.Item_Of(VERTEX_DEST);
      VERTEX_COMPARE(A_GRAPH.SOUR-
CE_ITEM.ID,VERTEX_SOURCE,SOURCE_FOUND);
      VERTEX_COMPARE(A_GRAPH.DES-
T_ITEM.ID,VERTEX_DEST,DEST_FOUND);
      if SOURCE_FOUND and DEST_FOUND then
        POINTS_FOUND := TRUE;
       else
        POINTS_FOUND := FALSE;
      end if;

    end FIND_ARC_POINTS;

--*************************************************************************
--This procedure determines the arcs of the graph.
--*************************************************************************
  procedure BUILD_ARC_LINK_LIST
                                    (WORKING_GRAPH,
                                      FULL_GRAPH     : in out TEST_GRAPH.GRAPH;
                                      ARC_LINK_LIST  : in out LINK) is

      ATTRIBUTE                : NAME;
      SOURCE,
      DESTINATION              : TEST_GRAPH.VERTEX;
      NEW_ARC                  : TEST_GRAPH.ARC;
      ARC_POINTS_FOUND,
      ARC_MATCH_MADE   : BOOLEAN;

    --*********************************************************************
    --This procedure traverses the arcs in the graph to determine the arcs in the graph.
```

```
--*********************************************************************
procedure BUILD_LIST(THE_ARC  : in TEST_GRAPH.ARC;
                     CONTINUE : out BOOLEAN) is

   AN_ARC : TEST_GRAPH.ARC := THE_ARC;

  begin
   ATTRIBUTE := TEST_GRAPH.Attribute_Of(AN_ARC);
   FIND_ARC_POINTS(AN_ARC,WORKING_GRAPH.ATTRIBUTE,
                   SOURCE.DESTINATION,ARC_POINTS_FOUND);
   if ARC_POINTS_FOUND then
    ARC_COMPARE(WORKING_GRAPH,SOURCE.DESTINATION,
                   ATTRIBUTE,NEW_ARC,ARC_MATCH_MADE);
    if not ARC_MATCH_MADE then
     ARC_LIST.INSERT_END(ARC_LINK_LIST, THE_ARC);
    end if;
   end if;
   CONTINUE := TRUE;
 end BUILD_LIST;
 procedure CREATE_ARC_LIST is new TEST_GRAPH.Iterate_Arcs(BUILD_LIST);


  begin     -- build_arc_link_list

   CREATE_ARC_LIST(FULL_GRAPH);

end BUILD_ARC_LINK_LIST;

--*********************************************************************
--This is the main procedure in the working package. This procedure is the driver to
--generate and calculate the cost of each combination of graphs based on the input set.
--*********************************************************************
 procedure generate_all_graphs_and_costs is

    -- A_GRAPH is a graph of the important nodes.
    -- FULL_GRAPH is a graph of all nodes and arcs.
    -- WORKING_GRAPH starts as a graph of all nodes only.


  FILE_NAME          : STRING(1..80) := (others => ' ');
  LAST               : INTEGER;
  ARC_COUNT          : NATURAL := 1;
  WORKING_GRAPH  : GRAPH;
```

```
--*********************************************************************
--This is the internal calling procedure to calculate the cost of each possible graph
--reachable based on the input set.
--*********************************************************************
procedure CALC_ARCS(ARC_LINK_LIST  : in out ARC_LIST.LINK;
                    WORKING_GRAPH  : in out TEST_GRAPH.GRAPH) is



  --*****************************************************************
  --This procedure Adds and arc to the graph. It first determines if the Source and
  --Destination are present.
  --*****************************************************************
  procedure ADD_ARC(AN_ARC_LIST : in out ARC_LIST.LINK;
                    TEMP_GRAPH  : in out TEST_GRAPH.GRAPH) is

    TEMP_ARC       : TEST_GRAPH.ARC;
    ATTRIBUTE      : NAME;
    SOURCE,
    DESTINATION    : TEST_GRAPH.VERTEX;
    ARC_POINTS_FOUND,
    ARC_MATCH_MADE  : BOOLEAN := FALSE;

   begin
    TEMP_ARC := AN_ARC_LIST.INFO;
    FIND_ARC_POINTS(TEMP_ARC,TEMP_GRAPH,ATTRIBUTE,SOURCE,
                    DESTINATION,ARC_POINTS_FOUND);
    if ARC_POINTS_FOUND then
     ARC_COMPARE(TEMP_GRAPH,SOURCE,DESTINATION,ATTRIBUTE,
                 TEMP_ARC,ARC_MATCH_MADE);
     if not ARC_MATCH_MADE then

TEST_GRAPH.Create(TEMP_ARC,ATTRIBUTE,SOURCE,DESTINATION,TEMP_G
RAPH);
      end if;
    end if;
  end ADD_ARC;
```

```
--*****************************************************************
--This procedure deletes the arcs of a graph. This procedure is used after the
--graph calculations are completed, prior to adding another node to the working
--graph.
--*****************************************************************
procedure DEL_ARC(ARC_LIST_HEAD  : in out ARC_LIST.LINK;
                  TEMP_GRAPH     : in out TEST_GRAPH.GRAPH) is

   TEMP_ARC       : TEST_GRAPH.ARC;
   ATTRIBUTE      : NAME;
   SOURCE,
   DESTINATION    : TEST_GRAPH.VERTEX;
   ARC_POINTS_FOUND,
   ARC_MATCH_MADE  : BOOLEAN := FALSE;

  begin
   TEMP_ARC :=  ARC_LIST_HEAD.INFO;
   FIND_ARC_POINTS(TEMP_ARC,TEMP_GRAPH,ATTRIBUTE,SOURCE,
                   DESTINATION,ARC_POINTS_FOUND);
   if ARC_POINTS_FOUND then
    ARC_COMPARE(TEMP_GRAPH,SOURCE,DESTINATION,ATTRIBUTE,
                TEMP_ARC,ARC_MATCH_MADE);
    if ARC_MATCH_MADE then
     TEST_GRAPH.Destroy(TEMP_ARC,TEMP_GRAPH);
    end if;
   end if;
 end DEL_ARC;


 begin   -- CALC ARCS
  if ARC_LINK_LIST /= null then
   ADD_ARC(ARC_LINK_LIST, WORKING_GRAPH);
   CALCULATE_GRAPH_COST(WORKING_GRAPH);
   CALC_ARCS(ARC_LINK_LIST.NEXT, WORKING_GRAPH);
   DEL_ARC(ARC_LINK_LIST, WORKING_GRAPH);
   CALC_ARCS(ARC_LINK_LIST.NEXT, WORKING_GRAPH);
  end if;

end CALC_ARCS;
```

```
--*********************************************************************
--This procedure traverses the vertices of the graph.
--*********************************************************************
procedure traverse_vertices(The_Vertex : in VERTEX;
                            CONTINUE   : out BOOLEAN) is

  TEMP_VERTEX,
  FOUND_VERTEX  : VERTEX;
  TEMP_ITEM     : ITEM;
  VERTEX_FOUND  : BOOLEAN := false;

 begin
  TEMP_ITEM := TEST_GRAPH.ITEM_OF(THE_VERTEX);
  VERTEX_COMPARE(WORKING_GRAPH,TEMP_ITEM.ID,
                          FOUND_VERTEX,VERTEX_FOUND);
  if not VERTEX_FOUND then
   TEST_GRAPH.Add(TEMP_VERTEX,TEMP_ITEM,WORKING_GRAPH);
   ARC_LIST.Clear(THE_ARC_LIST);
   CALCULATE_GRAPH_COST(WORKING_GRAPH);
        -- working graph starts with only the important nodes.
   BUILD_ARC_LINK_LIST(WORKING_GRAPH, FULL_GRAPH,
                          THE_ARC_LIST);
   CALC_ARCS(THE_ARC_LIST, WORKING_GRAPH);
  end if;
  if TEST_GRAPH.Number_Of_Vertices_In(WORKING_GRAPH) =
      TEST_GRAPH.Number_Of_Vertices_In(FULL_GRAPH) then
   CONTINUE := FALSE;
  else
   CONTINUE := TRUE;
  end if;
end traverse_vertices;
procedure Vertex_Traversal is new
        TEST_GRAPH.Iterate_Vertices(traverse_vertices);


begin           -- generate_all_graphs_and_costs.

 TEST_GRAPH.CLEAR(FINISH_GRAPH);
 TEST_GRAPH.CLEAR(WORKING_GRAPH);

 CREATE_DISCONNECTED_GRAPH(FULL_GRAPH, FINISH_GRAPH);
                -- FINISHED GRAPH is a graph of all nodes.
```

```
   TEST_GRAPH.Copy(IMPORTANT_GRAPH, WORKING_GRAPH);

-- for starting important graph.

   CALCULATE_GRAPH_COST(WORKING_GRAPH);
   BUILD_ARC_LINK_LIST(WORKING_GRAPH, FULL_GRAPH,
THE_ARC_LIST);
   CALC_ARCS(THE_ARC_LIST, WORKING_GRAPH);
   ARC_LIST.Clear(THE_ARC_LIST);      -- Starting Graph has been completed

-- for remainder of the graph.

   VERTEX_TRAVERSAL(FULL_GRAPH);   -- A vertex traversal of the graph
                        -- having all nodes and arcs.

 end generate_all_graphs_and_costs;

end GRAPH_WORKS;
```

APPENDIX C: Graph Package Specification

graph_spec

```
with Set_Package;
generic
   type Item      is private;
   type Attribute is private;
package Graph_Package is

   type Graph  is limited private;
   type Vertex is private;
   type Arc    is private;

   Null_Vertex : constant Vertex;
   Null_Arc    : constant Arc;

   procedure Copy        (From_The_Graph    : in     Graph;
                          To_The_Graph      : in out Graph);
   procedure Clear       (The_Graph         : in out Graph);
   procedure Add         (The_Vertex        : in out Vertex;
                          With_The_Item     : in     Item;
                          To_The_Graph      : in out Graph);
   procedure Remove      (The_Vertex        : in out Vertex;
                          From_The_Graph    : in out Graph);
   procedure Set_Item    (Of_The_Vertex     : in out Vertex;
                          To_The_Item       : in     Item);
   procedure Create      (The_Arc           : in out Arc;
                          With_The_Attribute : in    Attribute;
                          From_The_Vertex   : in out Vertex;
                          To_The_Vertex     : in     Vertex;
                          In_The_Graph      : in out Graph);
   procedure Destroy     (The_Arc           : in out Arc;
                          In_The_Graph      : in out Graph);
   procedure Set_Attribute (Of_The_Arc      : in out Arc;
                          To_The_Attribute  : in     Attribute);

   function Is_Empty           (The_Graph   : in Graph)  return Boolean;
   function Is_Null            (The_Vertex  : in Vertex) return Boolean;
   function Is_Null            (The_Arc     : in Arc)    return Boolean;
   function Number_Of_Vertices_In (The_Graph : in Graph)  return Natural;
```

```
   function Number_Of_Arcs_In   (The_Graph   : in Graph) return Natural;
   function Number_Of_Arcs_From (The_Vertex  : in Vertex) return Natural;
   function Item_Of             (The_Vertex  : in Vertex) return Item;
   function Attribute_Of        (The_Arc     : in Arc)    return Attribute;
   function Source_Of           (The_Arc     : in Arc)    return Vertex;
   function Destination_Of      (The_Arc     : in Arc)    return Vertex;
   function Is_A_Member         (The_Vertex  : in Vertex;
                                 Of_The_Graph : in Graph) return Boolean;
   function Is_A_Member         (The_Arc     : in Arc;
                                 Of_The_Graph : in Graph) return Boolean;

   generic
      with procedure Process (The_Vertex : in  Vertex;
                              Continue   : out Boolean);
   procedure Iterate_Vertices (Over_The_Graph : in out Graph);

   generic
      with procedure Process (The_Arc  : in  Arc;
                              Continue : out Boolean);
   procedure Iterate_Arcs (Over_The_Graph : in out Graph);

   generic
      with procedure Process (The_Arc  : in  Arc;
                              Continue : out Boolean);
   procedure Reiterate (Over_The_Vertex : in Vertex);

   Overflow             : exception;
   Vertex_Is_Null       : exception;
   Vertex_Is_Not_In_Graph : exception;
   Vertex_Has_References  : exception;
   Arc_Is_Null          : exception;
   Arc_Is_Not_In_Graph  : exception;

private
   type Vertex_Node;
   type Vertex is access Vertex_Node;
   package Vertex_Set is new
     Set_Package(Item => Vertex);
   type Arc_Node;
   type Arc is access Arc_Node;
   package Arc_Set is new
     Set_Package(Item => Arc);
   type Graph is
```

```
      record
         The_Vertices : Vertex_Set.Set;
         The_Arcs     : Arc_Set.Set;
      end record;
   Null_Vertex : constant Vertex := null;
   Null_Arc    : constant Arc := null;
end Graph_Package;
```

map_spec

```
generic
   type Domain is private;
   type Ranges is private;
   Number_Of_Buckets : in Positive;
   with function Hash_Of (The_Domain : in Domain) return Positive;
package Map_Package is

   type Map is limited private;

   procedure Copy   (From_The_Map  : in     Map;
                     To_The_Map    : in out Map);
   procedure Clear  (The_Map       : in out Map);
   procedure Bind   (The_Domain    : in     Domain;
                     And_The_Range : in     Ranges;
                     In_The_Map    : in out Map);
   procedure Unbind (The_Domain    : in     Domain;
                     In_The_Map    : in out Map);

   function Is_Equal  (Left      : in Map;
                       Right     : in Map) return Boolean;
   function Extent_Of (The_Map   : in Map) return Natural;
   function Is_Empty  (The_Map   : in Map) return Boolean;
   function Is_Bound  (The_Domain : in Domain;
                       In_The_Map : in Map) return Boolean;
   function Range_Of  (The_Domain : in Domain;
                       In_The_Map : in Map) return Ranges;

   generic
      with procedure Process (The_Domain : in  Domain;
                              The_Range  : in  Ranges;
                              Continue   : out Boolean);
```

```
   procedure Iterate (Over_The_Map : in Map);

   Overflow          : exception;
   Domain_Is_Not_Bound : exception;
   Multiple_Binding    : exception;

private
   type Node;
   type Structure is access Node;
   type Map is array (Positive range 1 .. Number_Of_Buckets) of Structure;
end Map_Package;
```

set_spec

```
generic
   type Item is private;
package Set_Package is

   type Set is limited private;

   procedure Copy        (From_The_Set : in     Set;
                          To_The_Set   : in out Set);
   procedure Clear       (The_Set      : in out Set);
   procedure Add         (The_Item     : in     Item;
                          To_The_Set   : in out Set);
   procedure Remove      (The_Item     : in     Item;
                          From_The_Set : in out Set);
   procedure Union       (Of_The_Set   : in     Set;
                          And_The_Set  : in     Set;
                          To_The_Set   : in out Set);
   procedure Intersection (Of_The_Set   : in     Set;
                          And_The_Set  : in     Set;
                          To_The_Set   : in out Set);
   procedure Difference  (Of_The_Set   : in     Set;
                          And_The_Set  : in     Set;
                          To_The_Set   : in out Set);

   function Is_Equal          (Left      : in Set;
                               Right     : in Set)  return Boolean;
   function Extent_Of         (The_Set   : in Set)  return Natural;
   function Is_Empty          (The_Set   : in Set)  return Boolean;
   function Is_A_Member       (The_Item  : in Item;
```

```
                    Of_The_Set : in Set) return Boolean;
  function Is_A_Subset      (Left  . : in Set;
                    Right     : in Set) return Boolean;
  function Is_A_Proper_Subset (Left     : in Set;
                    Right     : in Set) return Boolean;

  generic
    with procedure Process (The_Item : in  Item;
                    Continue : out Boolean);
  procedure Iterate (Over_The_Set : in Set);

  Overflow         : exception;
  Item_Is_In_Set   : exception;
  Item_Is_Not_In_Set : exception;

private
  type Node;
  type Set is access Node;
end Set_Package;
```

APPENDIX D: Program Variable Aggregates

FILENAME	ID	AGGREGATES	SEQUENCING
ability.2f	6	ability (13) multiaction smallaction	none
bkcalc.1f	35	bkcalk (47)	34
borders.2f	13	borders (20) whenborders	12
centerpt.1f	11	centerpt evenrows	99, 3
concurrent.1f	45	concurrent (108)	12, 16, 32, 40
constBatt.1f	5	constBatt (10)	1
contproc.2f	42	contproc (59) OffQueue HighFirstOffQ	41
cumulative.1f	10	cumulative	3
destrolle.1f	31	destrolle (38)	28
diffupdates.1f	8	diffupdates diffinc (15)	1
emax.1f	36	emax (62)	35
fcalc.2f	none	fcalc (48) nfmod fnocas (49) sumfmn (50)	none

FILENAME	ID	AGGREGATES	SEQUENCING
function.1f	9	function	1, 8
Heightcond.1f	17	Heightcond (23)	15, 16
Hgtcondzero.1f	18	hgtcondzero (24)	16
infrd.2f	44	infrd	41
kamod.1f	none	kamod (30)	31, 32
kfzero.2f	24	kfzero (29) kpzero TooFar (81)	1
minarmy.1f	4	minarmy (7) minBatt	1
mostweapons.2f	23	mostweapons (33) mosttargets (34) notjustonetarg (36)	22
neednewaxy.1f	22	neednewaxy (35)	19
nkmod.2f	25	nkmod (31)	24
nofrndfire.1f	30	nofrndfire (12)	19, 27
noobs.1f	19	noobs (51)	16
noobsnoatter.5f	29	noobsnoatter (28)	19, 27, 69
notargnofire.1f	30.5	notargnofire (37)	19, 27, 29
nsmod.2f	38	nsmod (54)	1
numrange.1f	7	numrange (14)	6

FILENAME	ID	AGGREGATES	SEQUENCING
outarmytoBatt.1f	2	outarmytoBatt (8)	1
procdest.2f	43	procdest (60)	41
reptinc.1f	41	reptinc (25) reptexpire (26) reptobsconf reptreptconf reptuse (78)	40
reptmdef.2f	40	reptmdef (51) whenrept (52) reptdest (53) commdes (55) Scomm (56) Srept (57) ImpleMess (61) CommDelay (102)	38
simpos.1f	33	simpos (16)	32, 16
simulationduration.2f	1	simulationduration (2) durationname (3) Durationrange (65)	none
solidterrain.2f	12	solidterrain (6) whenmove (17)	1, 70
srmin.2f	15	srmin (22)	none
terrweaeffect.2f	21	terrweaeffect (5)	12, 70, 69
umod.3f	28	umod (39) destrbatt (40) AllBatts (11)	1

FILENAME	ID	AGGREGATES	SEQUENCING
vgcalc.2f	14	vgcalc (19) whenvg	12, 70
whenattrit.1f	32	whenattrit (27) nointerattrit	1
whenku.2f	27	whenku (32)	26
whenobs.1f	16	whenobs (21)	1
whenRD.1f	39	whenRD mesgRD	1
whenrest.2f	37	whenrest (41) Dstrnotcas (44) ETooHigh (45) NolongerCAS (58) CASFixed (46)	34, 36
zeroduration.2f	3	zeroduration (4)	2

APPENDIX E: Input Data Sets

DATA_SET_ONE

```
suspicious 0  0.00   0.00
suspicious 1  26.00  9.00
suspicious 2  36.00  32.00
important  3   4.00  17.00
important  4  768.00  16.00
suspicious 5  42.00  26.00
important  6  30.00  13.00
important  7  225.00 41.00
suspicious 8  40.00  17.00
suspicious 9  90.00  33.00
important  10 101.00 55.00
important  11 20.00  41.00
important  12 221.00 76.00
important  13 54.00  64.00
important  14 210.00 37.00
suspicious 15 114.00 26.00
important  16 100.00 32.00
suspicious 17 126.00 12.00
important  18 46.00  7.00

TEST 01   0  9.00 1
TEST 12   1 55.00 2
TEST 214  2  5.00 14
TEST 13   1  8.00 3
TEST 16   1  7.00 6
TEST 09   0 33.00 9
TEST 910  9 21.00 10
TEST 17   1 33.00 7
TEST 15   1 14.00 5
TEST 512  5 44.00 12
TEST 511  5 12.00 11
TEST 18   1  8.00 8
TEST 815  8  7.00 15
TEST 1516 15 5.00 16
TEST 513  5 30.00 13
TEST 117  1  3.00 17
TEST 1718 17 7.00 18
TEST 14   1 13.00 4
```

DATA_SET_TWO

```
suspicious 0  0.00   0.00
important  1  26.00  9.00
suspicious 2  36.00  32.00
suspicious 3   4.00  17.00
suspicious 4  768.00  16.00
important  5  42.00  26.00
suspicious 6  30.00  13.00
suspicious 7  225.00 41.00
important  8  40.00  17.00
important  9  90.00  33.00
suspicious 10 101.00 55.00
suspicious 11 20.00  41.00
suspicious 12 221.00 76.00
suspicious 13 54.00  64.00
suspicious 14 210.00 37.00
suspicious 15 114.00 26.00
suspicious16 100.00  32.00
suspicious 17 126.00 12.00
suspicious 18 46.00  7.00

TEST 01   0  9.00 1
TEST 12   1 55.00
TEST 214  2  5.00 14
TEST 13   1  8.00 3
TEST 16   1  7.00 6
TEST 09   0 33.00 9
TEST 910  9 21.00 10
TEST 17   1 33.00 7
TEST 15   1 14.00 5
TEST 512  5 44.00 12
TEST 511  5 12.00 11
TEST 18   1  8.00 8
TEST 815  8  7.00 15
TEST 1516 15 5.00 16
TEST 513  5 30.00 13
TEST 117  1  3.00 17
TEST 1718 17 7.00 18
TEST 14   1 13.00 4
```

DATA_SET_THREE

```
important 0  0.00   0.00
important 1  26.00  9.00
suspicious 2  36.00  32.00
suspicious 3  4.00  17.00
suspicious 4  768.00  16.00
suspicious 5  42.00  26.00
suspicious 6  30.00  13.00
suspicious 7  225.00  41.00
suspicious 8  40.00  17.00
important 9  90.00  33.00
suspicious 10 101.00  55.00
suspicious 11  20.00  41.00
suspicious 12 221.00  76.00
suspicious 13  54.00  64.00
suspicious 14 210.00  37.00
suspicious 15 114.00  26.00
suspicious16 100.00  32.00
suspicious 17 126.00  12.00
suspicious 18  46.00  7.00

TEST 01  0  9.00 1
TEST 12  1 55.00 2
TEST 214  2  5.00 14
TEST 13  1  8.00 3
TEST 16  1  7.00 6
TEST 09  0 33.00 9
TEST 910  9 21.00 10
TEST 17  1 33.00 7
TEST 15  1 14.00 5
TEST 512  5 44.00 12
TEST 511  5 12.00 11
TEST 18  1  8.00 8
TEST 815  8  7.00 15
TEST 1516 15 5.00 16
TEST 513  5 30.00 13
TEST 117  1  3.00 17
TEST 1718 17 7.00 18
TEST 14  1 13.00 4
```

DATA_SET_FOUR

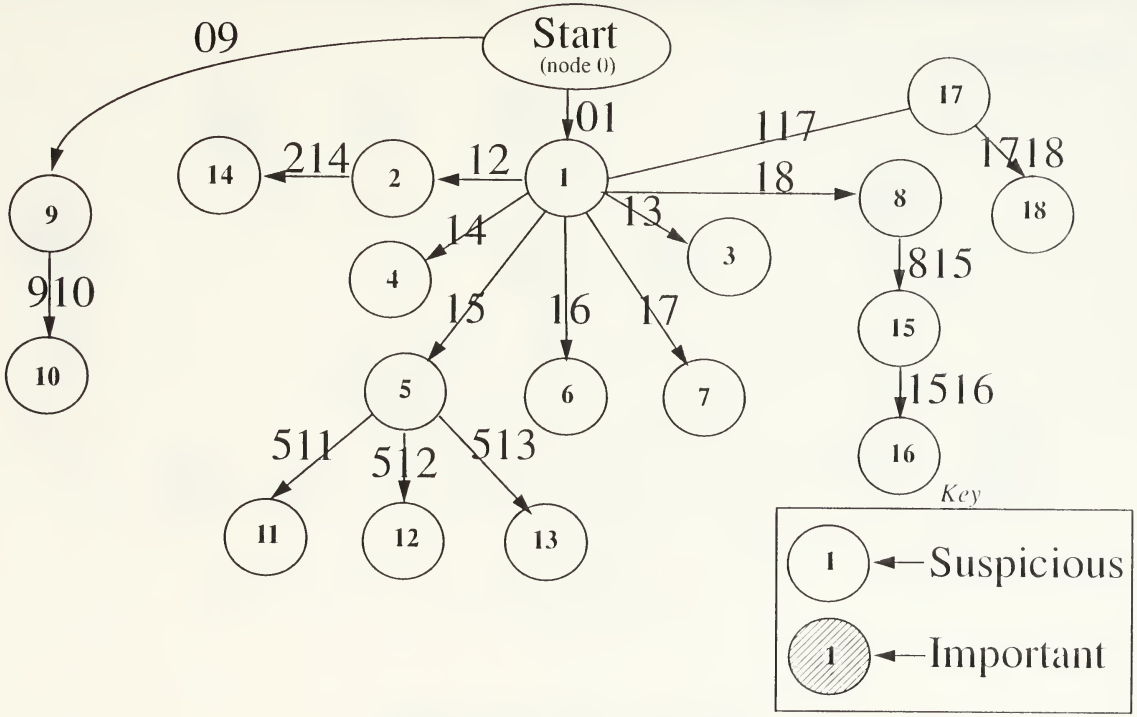
```
important 0  0.00   0.00
important 1  26.00  9.00
suspicious 2  36.00  32.00
suspicious 3  4.00  17.00
suspicious 4  768.00  16.00
suspicious 5  42.00  26.00
suspicious 6  30.00  13.00
suspicious 7  225.00  41.00
suspicious 8  40.00  17.00
important 9  90.00  33.00
suspicious 10 101.00  55.00
suspicious 11  20.00  41.00
suspicious 12 221.00  76.00
suspicious 13  54.00  64.00
suspicious 14 210.00  37.00
suspicious 15 114.00  26.00
suspicious16 100.00  32.00
suspicious 17 126.00  12.00
suspicious 18  46.00  7.00
suspicious 19  96.00  42.00

TEST 01  0  9.00 1
TEST 12  1 55.00 2
TEST 214  2  5.00 14
TEST 13  1  8.00 3
TEST 16  1  7.00 6
TEST 09  0 33.00 9
TEST 910  9 21.00 10
TEST 14  1 13.00 4
TEST 15  1 14.00 5
TEST 17  1 33.00 7
TEST 511  5 12.00 11
TEST 512  5 44.00 12
TEST 513  5 30.00 13
TEST 18  1  8.00 8
TEST 815  8  7.00 15
TEST 1516 15 5.00 16
TEST 1517 15 3.00 17
TEST 117  1  3.00 17
TEST 1718 17 7.00 18
```

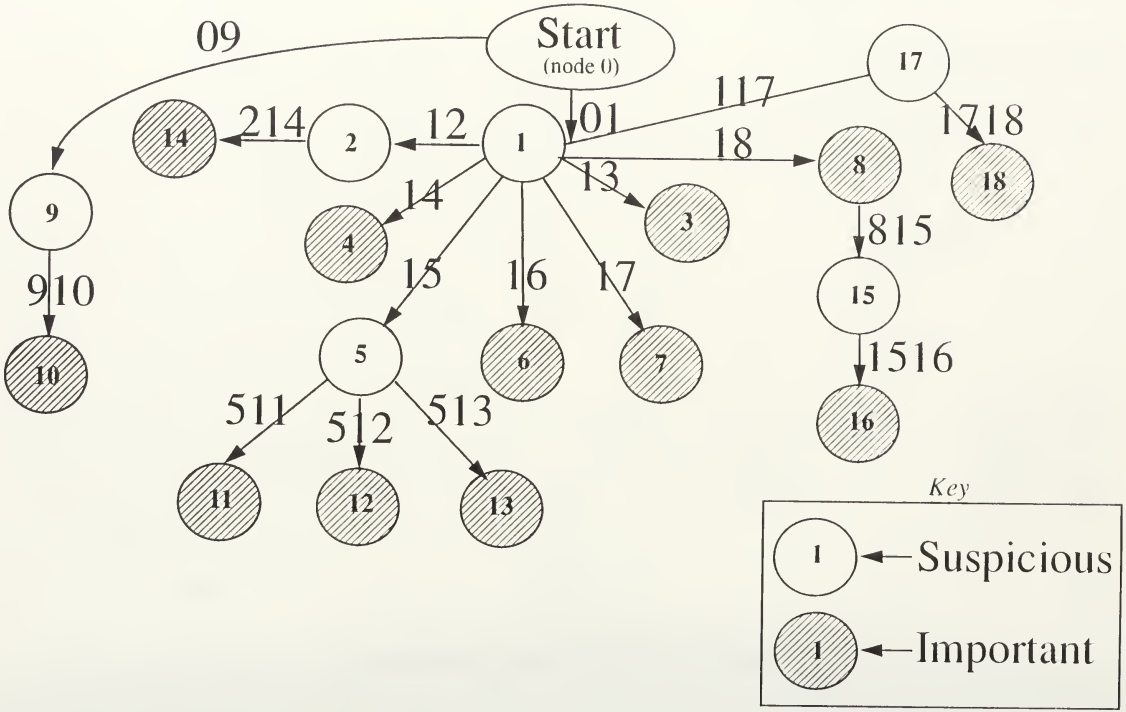
DATA_SET_FOUR

TEST 188 18 8.00 8
TEST 1519 15 8.00 19
TEST 1819 18 8.00 19

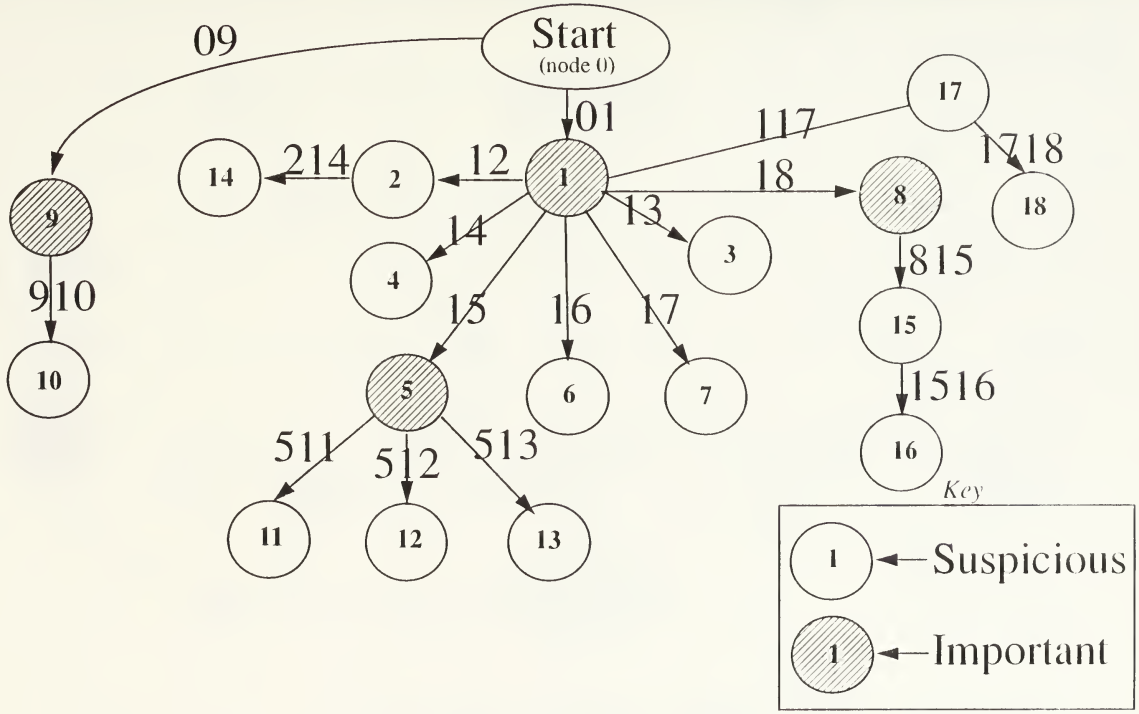
General Data Set Graph



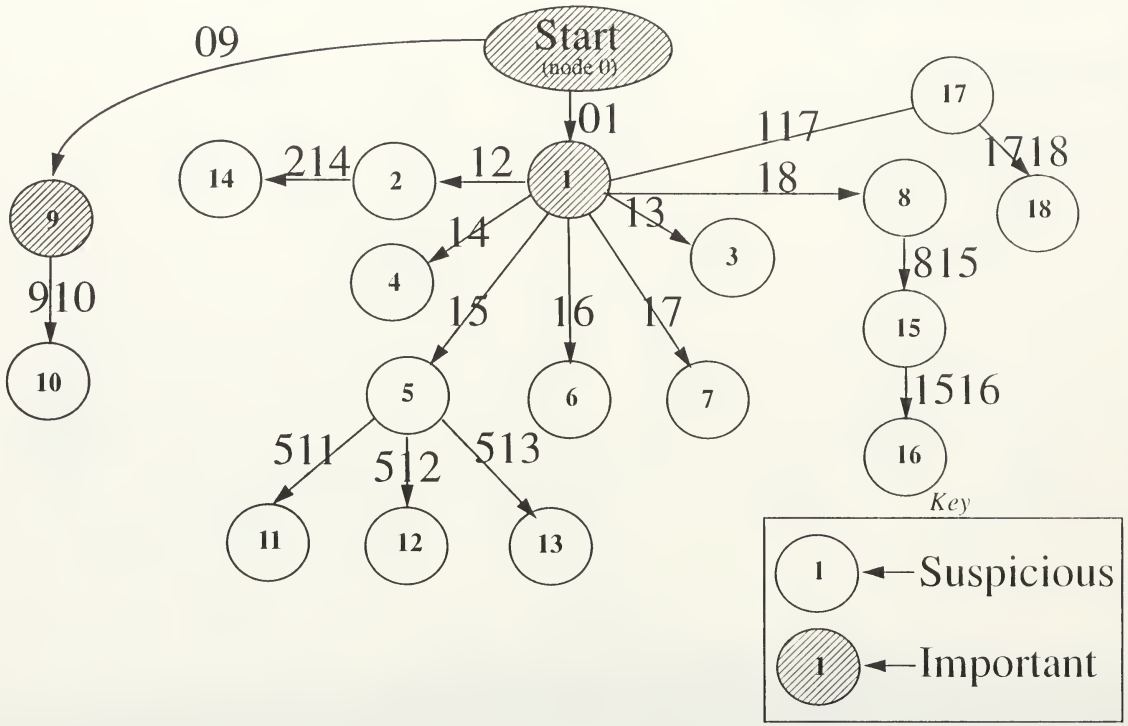
Data Set One Graph



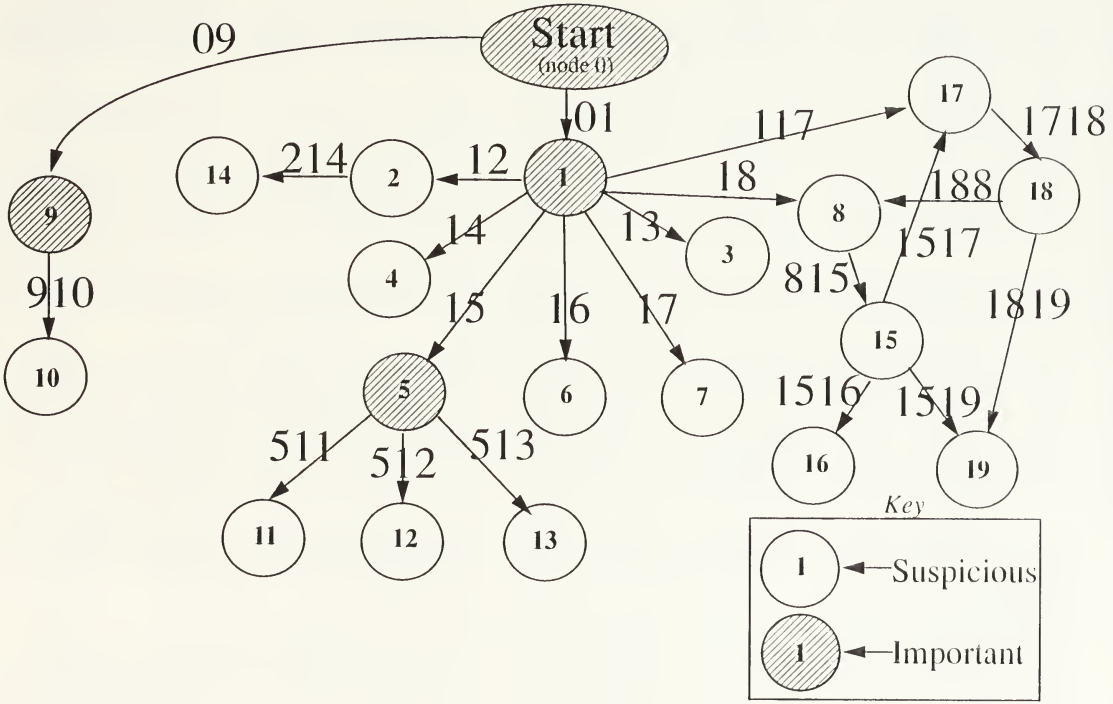
Data Set Two Graph



Data Set Three Graph



Data Set Four Graph



APPENDIX G: Scheduling Tool Output Data

```
**********************
*GRAPH_ONE_DATA_SUMMARY*
**********************
```

Num_Graphs	nds/arc	C(init) min	C(init) max	C(init) mean	C(init) std_dev	C(test) min	C(test) max	C(test) mean	C(test) std_dev	C(total) min	C(total) max	C(total) mean	C(total) std_dev
n = 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 1779.00)													
1.0	11/ 0	399.0	399.0	399.0	0.0	0.0	0.0	0.0	0.0	2178.0	2178.0	2178.0	0.0
Tot: 1.0		399.0	399.0	399.0	0.0	0.0	0.0	0.0	0.0	2178.0	2178.0	2178.0	0.0
n = 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 1779.00)													
1.0	12/ 0	399.0	399.0	399.0	0.0	0.0	0.0	0.0	0.0	2178.0	2178.0	2178.0	0.0
Tot: 1.0		399.0	399.0	399.0	0.0	0.0	0.0	0.0	0.0	2178.0	2178.0	2178.0	0.0
n = 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 1805.00)													
1.0	13/ 0	408.0	408.0	408.0	0.0	0.0	0.0	0.0	0.0	2213.0	2213.0	2213.0	0.0
5.0	13/ 1	367.0	399.0	388.8	11.3	7.0	33.0	14.0	9.7	2204.0	2213.0	2207.8	3.3
10.0	13/ 2	350.0	386.0	369.6	13.8	15.0	46.0	28.0	11.9	2196.0	2210.0	2202.6	4.1
10.0	13/ 3	334.0	370.0	350.4	13.8	24.0	55.0	42.0	11.9	2190.0	2204.0	2197.4	4.1
5.0	13/ 4	321.0	353.0	331.2	11.3	37.0	63.0	56.0	9.7	2187.0	2196.0	2192.2	3.3
1.0	13/ 5	312.0	312.0	312.0	0.0	70.0	70.0	70.0	0.0	2187.0	2187.0	2187.0	0.0
Tot: 32.0		312.0	408.0	360.0	24.9	0.0	70.0	35.0	19.1	2187.0	2213.0	2200.0	6.9
n = 2 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 1841.00)													
1.0	14/ 0	440.0	440.0	440.0	0.0	0.0	0.0	0.0	0.0	2281.0	2281.0	2281.0	0.0
7.0	14/ 1	399.0	431.0	416.4	11.8	5.0	55.0	18.6	17.3	2249.0	2304.0	2276.0	15.0
21.0	14/ 2	362.0	418.0	392.9	15.3	12.0	88.0	37.1	22.3	2240.0	2304.0	2271.0	19.3
35.0	14/ 3	330.0	402.0	369.3	16.7	20.0	101.0	55.7	24.4	2232.0	2301.0	2266.0	21.2
35.0	14/ 4	313.0	385.0	345.7	16.7	29.0	110.0	74.3	24.4	2226.0	2295.0	2261.0	21.2
21.0	14/ 5	297.0	353.0	322.1	15.3	42.0	118.0	92.9	22.3	2223.0	2287.0	2256.0	19.3
7.0	14/ 6	284.0	316.0	298.6	11.8	75.0	125.0	111.4	17.3	2223.0	2278.0	2251.0	15.0
1.0	14/ 7	275.0	275.0	275.0	0.0	130.0	130.0	130.0	0.0	2246.0	2246.0	2246.0	0.0
Tot: 128.0		275.0	440.0	357.5	34.9	0.0	130.0	65.0	33.5	2223.0	2304.0	2263.5	20.9
n = 5 2 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 1883.00)													
1.0	15/ 0	466.0	466.0	466.0	0.0	0.0	0.0	0.0	0.0	2349.0	2349.0	2349.0	0.0
11.0	15/ 1	390.0	457.0	432.2	20.3	5.0	55.0	20.9	16.2	2315.0	2372.0	2336.1	16.7
55.0	15/ 2	326.0	444.0	398.4	27.2	12.0	99.0	41.8	21.7	2283.0	2372.0	2323.2	22.4
165.0	15/ 3	285.0	428.0	364.5	31.4	20.0	132.0	62.7	25.0	2251.0	2369.0	2310.3	25.9
330.0	15/ 4	244.0	411.0	330.7	33.9	29.0	162.0	83.6	27.0	2222.0	2363.0	2297.4	28.0
462.0	15/ 5	207.0	385.0	296.9	35.1	41.0	176.0	104.5	28.0	2210.0	2355.0	2284.5	29.0
462.0	15/ 6	175.0	353.0	263.1	35.1	54.0	189.0	125.5	28.0	2201.0	2346.0	2271.5	29.0
330.0	15/ 7	149.0	316.0	229.3	33.9	68.0	201.0	146.4	27.0	2193.0	2334.0	2258.6	28.0
165.0	15/ 8	132.0	275.0	195.5	31.4	98.0	210.0	167.3	25.0	2187.0	2305.0	2245.7	25.9
55.0	15/ 9	116.0	234.0	161.6	27.2	131.0	218.0	188.2	21.7	2184.0	2273.0	2232.8	22.4
11.0	15/10	103.0	170.0	127.8	20.3	175.0	225.0	209.1	16.2	2184.0	2241.0	2220.0	16.7
1.0	15/11	94.0	94.0	94.0	0.0	230.0	230.0	230.0	0.0	2207.0	2207.0	2207.0	0.0
Tot: 2048.0		94.0	466.0	280.0	65.4	0.0	230.0	115.0	43.8	2184.0	2372.0	2278.0	35.0
n = 8 5 2 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 1923.00)													
1.0	16/ 0	483.0	483.0	483.0	0.0	0.0	0.0	0.0	0.0	2406.0	2406.0	2406.0	0.0
12.0	16/ 1	407.0	474.0	450.6	20.0	5.0	55.0	19.8	15.9	2372.0	2429.0	2393.4	16.0
66.0	16/ 2	343.0	461.0	418.2	26.9	12.0	99.0	39.7	21.4	2340.0	2429.0	2380.8	21.6
220.0	16/ 3	302.0	445.0	385.8	31.3	20.0	132.0	59.5	24.9	2308.0	2426.0	2368.3	25.1
495.0	16/ 4	261.0	428.0	353.3	34.0	28.0	162.0	79.3	27.1	2279.0	2420.0	2355.7	27.4
792.0	16/ 5	224.0	411.0	320.9	35.6	37.0	176.0	99.2	28.3	2267.0	2412.0	2343.1	28.6
924.0	16/ 6	192.0	385.0	288.5	36.1	49.0	189.0	119.0	28.7	2258.0	2403.0	2330.5	29.0
792.0	16/ 7	166.0	353.0	256.1	35.6	62.0	201.0	138.8	28.3	2249.0	2394.0	2317.9	28.6
495.0	16/ 8	149.0	316.0	223.7	34.0	76.0	210.0	158.7	27.1	2241.0	2382.0	2305.3	27.4
220.0	16/ 9	132.0	275.0	191.3	31.3	106.0	218.0	178.5	24.9	2235.0	2353.0	2292.8	25.1
66.0	16/10	116.0	234.0	158.8	26.9	139.0	226.0	198.3	21.4	2232.0	2321.0	2280.2	21.6
12.0	16/11	103.0	170.0	126.4	20.0	183.0	233.0	218.2	15.9	2232.0	2289.0	2267.6	16.0
1.0	16/12	94.0	94.0	94.0	0.0	238.0	238.0	238.0	0.0	2255.0	2255.0	2255.0	0.0
Tot: 4096.0		94.0	483.0	288.5	65.9	0.0	238.0	119.0	44.0	2232.0	2429.0	2330.5	35.3

n = 9 8 5 2 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 2013.00)

1.0	17/ 0	516.0	516.0	516.0	0.0	0.0	0.0	0.0	0.0	2529.0	2529.0	2529.0	0.0
14.0	17/ 1	440.0	507.0	481.9	19.4	5.0	55.0	20.9	15.1	2495.0	2552.0	2515.8	16.3
91.0	17/ 2	376.0	494.0	447.9	26.3	12.0	99.0	41.7	20.5	2461.0	2552.0	2502.6	22.1
364.0	17/ 3	321.0	478.0	413.8	30.9	20.0	132.0	62.6	24.0	2429.0	2552.0	2489.4	25.9
1001.0	17/ 4	280.0	461.0	379.7	34.0	28.0	165.0	83.4	26.4	2397.0	2549.0	2476.1	28.5
2002.0	17/ 5	239.0	444.0	345.6	36.0	37.0	195.0	104.3	28.0	2368.0	2543.0	2462.9	30.2
3003.0	17/ 6	202.0	418.0	311.6	37.2	49.0	216.0	125.1	29.0	2356.0	2535.0	2449.7	31.2
3432.0	17/ 7	169.0	386.0	277.5	37.6	62.0	230.0	146.0	29.3	2347.0	2526.0	2436.5	31.6
3003.0	17/ 8	137.0	353.0	243.4	37.2	76.0	243.0	166.9	29.0	2338.0	2517.0	2423.3	31.2
2002.0	17/ 9	111.0	316.0	209.4	36.0	97.0	255.0	187.7	28.0	2330.0	2505.0	2410.1	30.2
1001.0	17/10	94.0	275.0	175.3	34.0	127.0	264.0	208.6	26.4	2324.0	2476.0	2396.9	28.5
364.0	17/11	77.0	234.0	141.2	30.9	160.0	272.0	229.4	24.0	2321.0	2444.0	2383.6	25.9
91.0	17/12	61.0	179.0	107.1	26.3	193.0	280.0	250.3	20.5	2321.0	2412.0	2370.4	22.1
14.0	17/13	48.0	115.0	73.1	19.4	237.0	287.0	271.1	15.1	2321.0	2378.0	2357.2	16.3
1.0	17/14	39.0	39.0	39.0	0.0	292.0	292.0	292.0	0.0	2344.0	2344.0	2344.0	0.0
Tot: 16384.0		39.0	516.0	277.5	73.3	0.0	292.0	146.0	48.1	2321.0	2552.0	2436.5	39.2

n = 15 9 8 5 2 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 2127.00)

1.0	18/ 0	542.0	542.0	542.0	0.0	0.0	0.0	0.0	0.0	2669.0	2669.0	2669.0	0.0
16.0	18/ 1	466.0	533.0	508.6	18.2	5.0	55.0	19.0	14.9	2635.0	2692.0	2654.6	15.6
120.0	18/ 2	402.0	520.0	475.1	24.9	10.0	99.0	38.0	20.4	2601.0	2692.0	2640.1	21.3
560.0	18/ 3	347.0	504.0	441.7	29.4	17.0	132.0	57.0	24.1	2569.0	2692.0	2625.7	25.2
1820.0	18/ 4	306.0	487.0	408.3	32.6	24.0	165.0	76.0	26.7	2537.0	2689.0	2611.3	27.9
4368.0	18/ 5	265.0	470.0	374.8	34.9	32.0	195.0	95.0	28.6	2508.0	2683.0	2596.8	29.9
8008.0	18/ 6	228.0	444.0	341.4	36.4	40.0	216.0	114.0	29.9	2481.0	2675.0	2582.4	31.2
11440.0	18/ 7	195.0	418.0	307.9	37.3	49.0	230.0	133.0	30.6	2462.0	2666.0	2567.9	32.0
12870.0	18/ 8	163.0	386.0	274.5	37.6	61.0	243.0	152.0	30.9	2450.0	2657.0	2553.5	32.2
11440.0	18/ 9	131.0	354.0	241.1	37.3	74.0	255.0	171.0	30.6	2441.0	2645.0	2539.1	32.0
8008.0	18/10	105.0	321.0	207.6	36.4	88.0	264.0	190.0	29.9	2432.0	2626.0	2524.6	31.2
4368.0	18/11	79.0	284.0	174.2	34.9	109.0	272.0	209.0	28.6	2424.0	2599.0	2510.2	29.9
1820.0	18/12	62.0	243.0	140.8	32.6	139.0	280.0	228.0	26.7	2418.0	2570.0	2495.8	27.9
560.0	18/13	45.0	202.0	107.3	29.4	172.0	287.0	247.0	24.1	2415.0	2538.0	2481.3	25.2
120.0	18/14	29.0	147.0	73.9	24.9	205.0	294.0	266.0	20.4	2415.0	2506.0	2466.9	21.3
16.0	18/15	16.0	83.0	40.4	18.2	249.0	299.0	285.0	14.9	2415.0	2472.0	2452.4	15.6
1.0	18/16	7.0	7.0	7.0	0.0	304.0	304.0	304.0	0.0	2438.0	2438.0	2438.0	0.0
Tot: 65536.0		7.0	542.0	274.5	76.2	0.0	304.0	152.0	48.3	2415.0	2692.0	2553.5	42.5

n = 17 15 9 8 5 2 1 0 18 16 14 13 12 11 10 7 6 4 3 (Eval Cost :: 2253.00)

1.0	19/ 0	554.0	554.0	554.0	0.0	0.0	0.0	0.0	0.0	2807.0	2807.0	2807.0	0.0
18.0	19/ 1	478.0	547.0	523.2	18.8	3.0	55.0	17.4	14.8	2773.0	2830.0	2793.7	15.1
153.0	19/ 2	414.0	538.0	492.4	25.8	8.0	99.0	34.9	20.3	2739.0	2830.0	2780.3	20.7
816.0	19/ 3	359.0	526.0	461.7	30.5	13.0	132.0	52.3	24.0	2707.0	2830.0	2767.0	24.6
3060.0	19/ 4	318.0	513.0	430.9	34.1	20.0	165.0	69.8	26.8	2675.0	2830.0	2753.7	27.4
8568.0	19/ 5	277.0	497.0	400.1	36.7	27.0	195.0	87.2	28.9	2646.0	2827.0	2740.3	29.6
18564.0	19/ 6	240.0	480.0	369.3	38.6	34.0	216.0	104.7	30.4	2619.0	2821.0	2727.0	31.1
31824.0	19/ 7	207.0	463.0	338.6	40.0	42.0	230.0	122.1	31.4	2600.0	2813.0	2713.7	32.2
43758.0	19/ 8	175.0	437.0	307.8	40.7	50.0	243.0	139.6	32.0	2588.0	2804.0	2700.3	32.8
48620.0	19/ 9	143.0	411.0	277.0	41.0	59.0	255.0	157.0	32.2	2579.0	2795.0	2687.0	33.0
43758.0	19/10	117.0	379.0	246.2	40.7	71.0	264.0	174.4	32.0	2570.0	2786.0	2673.7	32.8
31824.0	19/11	91.0	347.0	215.4	40.0	84.0	272.0	191.9	31.4	2561.0	2774.0	2660.3	32.2
18564.0	19/12	74.0	314.0	184.7	38.6	98.0	280.0	209.3	30.4	2553.0	2755.0	2647.0	31.1
8568.0	19/13	57.0	277.0	153.9	36.7	119.0	287.0	226.8	28.9	2547.0	2728.0	2633.7	29.6
3060.0	19/14	41.0	236.0	123.1	34.1	149.0	294.0	244.2	26.8	2544.0	2699.0	2620.3	27.4
816.0	19/15	28.0	195.0	92.3	30.5	182.0	301.0	261.7	24.0	2544.0	2667.0	2607.0	24.6
153.0	19/16	16.0	140.0	61.6	25.8	215.0	306.0	279.1	20.3	2544.0	2635.0	2593.7	20.7
18.0	19/17	7.0	76.0	30.8	18.8	259.0	311.0	296.6	14.8	2544.0	2601.0	2580.3	15.1
1.0	19/18	0.0	0.0	0.0	0.0	314.0	314.0	314.0	0.0	2567.0	2567.0	2567.0	0.0
Tot: 262144.0		0.0	554.0	277.0	76.5	0.0	314.0	157.0	48.5	2544.0	2830.0	2687.0	42.8

****
GRAPH_TWO_DATA_SUMMARY
****

C(init) C(test) C(total)

Num_Graphs	nds/arc	min	max	mean	std_dev	min	max	mean	std_dev	min	max	mean	std_dev
n = 9 8 5 1 (Eval Cost :: 198.00)													
1.0	4/ 0	85.0	85.0	85.0	0.0	0.0	0.0	0.0	0.0	283.0	283.0	283.0	0.0
2.0	4/ 1	59.0	68.0	63.5	4.5	8.0	14.0	11.0	3.0	271.0	274.0	272.5	1.5
1.0	4/ 2	42.0	42.0	42.0	0.0	22.0	22.0	22.0	0.0	262.0	262.0	262.0	0.0
Tot: 4.0		42.0	85.0	63.5	15.5	0.0	22.0	11.0	8.1	262.0	283.0	272.5	7.5
n = 0 9 8 5 1 (Eval Cost :: 198.00)													
1.0	5/ 0	85.0	85.0	85.0	0.0	0.0	0.0	0.0	0.0	283.0	283.0	283.0	0.0
4.0	5/ 1	52.0	76.0	63.8	9.1	8.0	33.0	16.0	10.1	271.0	283.0	277.8	5.4
6.0	5/ 2	26.0	59.0	42.5	10.5	17.0	47.0	32.0	11.6	262.0	283.0	272.5	6.2
4.0	5/ 3	9.0	33.0	21.3	9.1	31.0	56.0	48.0	10.1	262.0	274.0	267.3	5.4
1.0	5/ 4	0.0	0.0	0.0	0.0	64.0	64.0	64.0	0.0	262.0	262.0	262.0	0.0
Tot: 16.0		0.0	85.0	42.5	23.1	0.0	64.0	32.0	18.9	262.0	283.0	272.5	7.5
n = 2 0 9 8 5 1 (Eval Cost :: 234.00)													
1.0	6/ 0	117.0	117.0	117.0	0.0	0.0	0.0	0.0	0.0	351.0	351.0	351.0	0.0
5.0	6/ 1	84.0	108.0	93.6	9.2	8.0	55.0	23.8	18.0	339.0	374.0	351.4	12.3
10.0	6/ 2	52.0	91.0	70.2	11.2	17.0	88.0	47.6	22.1	330.0	374.0	351.8	15.0
10.0	6/ 3	26.0	65.0	46.8	11.2	31.0	102.0	71.4	22.1	330.0	374.0	352.2	15.0
5.0	6/ 4	9.0	33.0	23.4	9.2	64.0	111.0	95.2	18.0	330.0	365.0	352.6	12.3
1.0	6/ 5	0.0	0.0	0.0	0.0	119.0	119.0	119.0	0.0	353.0	353.0	353.0	0.0
Tot: 32.0		0.0	117.0	58.5	28.1	0.0	119.0	59.5	33.4	330.0	374.0	352.0	13.7
n = 3 2 0 9 8 5 1 (Eval Cost :: 238.00)													
1.0	7/ 0	134.0	134.0	134.0	0.0	0.0	0.0	0.0	0.0	372.0	372.0	372.0	0.0
6.0	7/ 1	101.0	125.0	111.7	8.7	8.0	55.0	21.2	17.5	360.0	395.0	370.8	11.7
15.0	7/ 2	69.0	108.0	89.3	11.0	16.0	88.0	42.3	22.1	351.0	395.0	369.7	14.8
20.0	7/ 3	43.0	91.0	67.0	11.7	25.0	102.0	63.5	23.4	342.0	395.0	368.5	15.8
15.0	7/ 4	26.0	65.0	44.7	11.0	39.0	111.0	84.7	22.1	342.0	386.0	367.3	14.8
6.0	7/ 5	9.0	33.0	22.3	8.7	72.0	119.0	105.8	17.5	342.0	377.0	366.2	11.7
1.0	7/ 6	0.0	0.0	0.0	0.0	127.0	127.0	127.0	0.0	365.0	365.0	365.0	0.0
Tot: 64.0		0.0	134.0	67.0	29.4	0.0	127.0	63.5	33.6	342.0	395.0	368.5	14.4
n = 4 3 2 0 9 8 5 1 (Eval Cost :: 1006.00)													
1.0	8/ 0	150.0	150.0	150.0	0.0	0.0	0.0	0.0	0.0	1156.0	1156.0	1156.0	0.0
7.0	8/ 1	117.0	141.0	128.6	8.4	8.0	55.0	20.0	16.4	1144.0	1179.0	1154.6	10.9
21.0	8/ 2	85.0	125.0	107.1	10.8	16.0	88.0	40.0	21.2	1135.0	1179.0	1153.1	14.1
35.0	8/ 3	59.0	108.0	85.7	11.8	25.0	102.0	60.0	23.2	1126.0	1179.0	1151.7	15.4
35.0	8/ 4	42.0	91.0	64.3	11.8	38.0	115.0	80.0	23.2	1123.0	1176.0	1150.3	15.4
21.0	8/ 5	25.0	65.0	42.9	10.8	52.0	124.0	100.0	21.2	1123.0	1167.0	1148.9	14.1
7.0	8/ 6	9.0	33.0	21.4	8.4	85.0	132.0	120.0	16.4	1123.0	1158.0	1147.4	10.9
1.0	8/ 7	0.0	0.0	0.0	0.0	140.0	140.0	140.0	0.0	1146.0	1146.0	1146.0	0.0
Tot: 128.0		0.0	150.0	75.0	30.4	0.0	140.0	70.0	34.2	1123.0	1179.0	1151.0	14.5
n = 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1036.00)													
1.0	9/ 0	163.0	163.0	163.0	0.0	0.0	0.0	0.0	0.0	1199.0	1199.0	1199.0	0.0
8.0	9/ 1	130.0	154.0	142.6	8.3	7.0	55.0	18.4	16.0	1187.0	1222.0	1197.0	10.3
28.0	9/ 2	98.0	141.0	122.3	10.9	15.0	88.0	36.8	20.9	1178.0	1222.0	1195.0	13.5
56.0	9/ 3	72.0	125.0	101.9	12.2	23.0	102.0	55.1	23.4	1169.0	1222.0	1193.0	15.1
70.0	9/ 4	55.0	108.0	81.5	12.6	32.0	115.0	73.5	24.1	1163.0	1219.0	1191.0	15.6
56.0	9/ 5	38.0	91.0	61.1	12.2	45.0	124.0	91.9	23.4	1160.0	1213.0	1189.0	15.1
28.0	9/ 6	22.0	65.0	40.8	10.9	59.0	132.0	110.3	20.9	1160.0	1204.0	1187.0	13.5
8.0	9/ 7	9.0	33.0	20.4	8.3	92.0	140.0	128.6	16.0	1160.0	1195.0	1185.0	10.3
1.0	9/ 8	0.0	0.0	0.0	0.0	147.0	147.0	147.0	0.0	1183.0	1183.0	1183.0	0.0
Tot: 256.0		0.0	163.0	81.5	31.1	0.0	147.0	73.5	34.4	1160.0	1222.0	1191.0	14.8
n = 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1261.00)													
1.0	10/ 0	204.0	204.0	204.0	0.0	0.0	0.0	0.0	0.0	1465.0	1465.0	1465.0	0.0
9.0	10/ 1	163.0	195.0	181.3	10.2	7.0	55.0	20.0	15.7	1453.0	1488.0	1462.3	9.9
36.0	10/ 2	130.0	182.0	158.7	13.4	15.0	88.0	40.0	20.8	1444.0	1488.0	1459.7	13.1
84.0	10/ 3	98.0	166.0	136.0	15.2	23.0	121.0	60.0	23.6	1435.0	1488.0	1457.0	14.8
126.0	10/ 4	72.0	149.0	113.3	16.1	32.0	135.0	80.0	24.9	1427.0	1485.0	1454.3	15.6
126.0	10/ 5	55.0	132.0	90.7	16.1	45.0	148.0	100.0	24.9	1421.0	1479.0	1451.7	15.6

84.0	10/ 6	38.0	106.0	68.0	15.2	59.0	157.0	120.0	23.6	1418.0	1471.0	1449.0	14.8
36.0	10/ 7	22.0	74.0	45.3	13.4	92.0	165.0	140.0	20.8	1418.0	1462.0	1446.3	13.1
9.0	10/ 8	9.0	41.0	22.7	10.2	125.0	173.0	160.0	15.7	1418.0	1453.0	1443.7	9.9
1.0	10/ 9	0.0	0.0	0.0	0.0	180.0	180.0	180.0	0.0	1441.0	1441.0	1441.0	0.0
Tot: 512.0		0.0	204.0	102.0	37.3	0.0	180.0	90.0	38.2	1418.0	1488.0	1453.0	15.4

n = 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1362.00)

1.0	11/ 0	259.0	259.0	259.0	0.0	0.0	0.0	0.0	0.0	1621.0	1621.0	1621.0	0.0
10.0	11/ 1	204.0	250.0	233.1	13.7	7.0	55.0	20.1	14.9	1587.0	1644.0	1615.2	13.3
45.0	11/ 2	163.0	237.0	207.2	18.2	15.0	88.0	40.2	19.9	1575.0	1644.0	1609.4	17.7
120.0	11/ 3	130.0	221.0	181.3	20.9	23.0	121.0	60.3	22.8	1566.0	1644.0	1603.6	20.3
210.0	11/ 4	98.0	204.0	155.4	22.3	32.0	142.0	80.4	24.4	1557.0	1641.0	1597.8	21.7
252.0	11/ 5	72.0	187.0	129.5	22.8	45.0	156.0	100.5	24.9	1549.0	1635.0	1592.0	22.1
210.0	11/ 6	55.0	161.0	103.6	22.3	59.0	169.0	120.6	24.4	1543.0	1627.0	1586.2	21.7
120.0	11/ 7	38.0	129.0	77.7	20.9	80.0	178.0	140.7	22.8	1540.0	1618.0	1580.4	20.3
45.0	11/ 8	22.0	96.0	51.8	18.2	113.0	186.0	160.8	19.9	1540.0	1609.0	1574.6	17.7
10.0	11/ 9	9.0	55.0	25.9	13.7	146.0	194.0	180.9	14.9	1540.0	1597.0	1568.8	13.3
1.0	11/10	0.0	0.0	0.0	0.0	201.0	201.0	201.0	0.0	1563.0	1563.0	1563.0	0.0
Tot: 1024.0		0.0	259.0	129.5	46.3	0.0	201.0	100.5	39.6	1540.0	1644.0	1592.0	22.9

n = 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1382.00)

1.0	12/ 0	300.0	300.0	300.0	0.0	0.0	0.0	0.0	0.0	1682.0	1682.0	1682.0	0.0
11.0	12/ 1	245.0	291.0	272.7	13.7	7.0	55.0	19.4	14.4	1648.0	1705.0	1674.1	14.3
55.0	12/ 2	204.0	278.0	245.5	18.4	15.0	88.0	38.7	19.3	1619.0	1705.0	1666.2	19.2
165.0	12/ 3	163.0	262.0	218.2	21.3	23.0	121.0	58.1	22.3	1607.0	1705.0	1658.3	22.2
330.0	12/ 4	130.0	245.0	190.9	23.0	32.0	142.0	77.5	24.1	1598.0	1702.0	1650.4	24.0
462.0	12/ 5	98.0	228.0	163.6	23.8	44.0	156.0	96.8	25.0	1589.0	1696.0	1642.5	24.8
462.0	12/ 6	72.0	202.0	136.4	23.8	57.0	169.0	116.2	25.0	1581.0	1688.0	1634.5	24.8
330.0	12/ 7	55.0	170.0	109.1	23.0	71.0	181.0	135.5	24.1	1575.0	1679.0	1626.6	24.0
165.0	12/ 8	38.0	137.0	81.8	21.3	92.0	190.0	154.9	22.3	1572.0	1670.0	1618.7	22.2
55.0	12/ 9	22.0	96.0	54.5	18.4	125.0	198.0	174.3	19.3	1572.0	1658.0	1610.8	19.2
11.0	12/10	9.0	55.0	27.3	13.7	158.0	206.0	193.6	14.4	1572.0	1629.0	1602.9	14.3
1.0	12/11	0.0	0.0	0.0	0.0	213.0	213.0	213.0	0.0	1595.0	1595.0	1595.0	0.0
Tot: 2048.0		0.0	300.0	150.0	50.6	0.0	213.0	106.5	40.0	1572.0	1705.0	1638.5	27.1

n = 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1603.00)

1.0	13/ 0	376.0	376.0	376.0	0.0	0.0	0.0	0.0	0.0	1979.0	1979.0	1979.0	0.0
12.0	13/ 1	300.0	367.0	344.7	18.8	7.0	55.0	21.4	15.4	1945.0	2002.0	1969.1	15.2
66.0	13/ 2	245.0	354.0	313.3	25.4	15.0	99.0	42.8	20.8	1913.0	2002.0	1959.2	20.5
220.0	13/ 3	204.0	338.0	282.0	29.5	23.0	132.0	64.3	24.1	1884.0	2002.0	1949.3	23.9
495.0	13/ 4	163.0	321.0	250.7	32.1	32.0	165.0	85.7	26.3	1872.0	2000.0	1939.3	26.0
792.0	13/ 5	130.0	304.0	219.3	33.6	44.0	186.0	107.1	27.5	1863.0	1993.0	1929.4	27.2
924.0	13/ 6	98.0	278.0	188.0	34.1	57.0	200.0	128.5	27.8	1854.0	1985.0	1919.5	27.6
792.0	13/ 7	72.0	246.0	156.7	33.6	71.0	213.0	150.0	27.5	1846.0	1976.0	1909.6	27.2
495.0	13/ 8	55.0	213.0	125.3	32.1	92.0	225.0	171.3	26.3	1840.0	1967.0	1900.0	26.0
220.0	13/ 9	38.0	172.0	94.0	29.5	125.0	234.0	192.8	24.1	1837.0	1955.0	1889.8	23.9
66.0	13/10	22.0	131.0	62.7	25.4	158.0	242.0	214.2	20.8	1837.0	1926.0	1879.8	20.5
12.0	13/11	9.0	76.0	31.3	18.8	202.0	250.0	235.6	15.4	1837.0	1894.0	1870.0	15.2
1.0	13/12	0.0	0.0	0.0	0.0	257.0	257.0	257.0	0.0	1860.0	1860.0	1860.0	0.0
Tot: 4096.0		0.0	376.0	188.0	63.3	0.0	257.0	128.5	45.7	1837.0	2002.0	1919.5	31.5

n = 13 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1657.00)

1.0	14/ 0	440.0	440.0	440.0	0.0	0.0	0.0	0.0	0.0	2097.0	2097.0	2097.0	0.0
13.0	14/ 1	364.0	431.0	406.2	20.1	7.0	55.0	22.1	15.0	2063.0	2120.0	2085.2	16.0
78.0	14/ 2	300.0	418.0	372.3	27.2	15.0	99.0	44.2	20.3	2029.0	2120.0	2073.5	21.6
286.0	14/ 3	245.0	402.0	338.5	31.7	23.0	132.0	66.2	23.7	2000.0	2120.0	2061.7	25.3
715.0	14/ 4	204.0	385.0	304.6	34.8	32.0	165.0	88.3	25.9	1968.0	2117.0	2050.0	27.7
1287.0	14/ 5	163.0	368.0	270.8	36.7	44.0	195.0	110.4	27.3	1956.0	2111.0	2038.2	29.2
1716.0	14/ 6	130.0	342.0	236.9	37.6	57.0	216.0	132.5	28.0	1947.0	2103.0	2026.4	29.9
1716.0	14/ 7	98.0	310.0	203.1	37.6	71.0	230.0	154.5	28.0	1938.0	2094.0	2014.6	29.9
1287.0	14/ 8	72.0	277.0	169.2	36.7	92.0	243.0	176.6	27.3	1930.0	2085.0	2002.8	29.2
715.0	14/ 9	55.0	236.0	135.4	34.8	122.0	255.0	198.7	25.9	1924.0	2073.0	1991.1	27.7
286.0	14/10	38.0	195.0	101.5	31.7	155.0	264.0	220.8	23.7	1921.0	2044.0	1979.3	25.3
78.0	14/11	22.0	140.0	67.7	27.2	188.0	272.0	242.8	20.3	1921.0	2012.0	1967.5	21.6
13.0	14/12	9.0	76.0	33.8	20.1	232.0	280.0	264.9	15.0	1921.0	1978.0	1955.8	16.0
1.0	14/13	0.0	0.0	0.0	0.0	287.0	287.0	287.0	0.0	1944.0	1944.0	1944.0	0.0

Tot:	8192.0	0.0	440.0	220.0	70.9	0.0	287.0	143.5	48.1	1921.0	2120.0	2020.5	35.8

n = 14 13 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1867.00)

1.0	15/ 0	477.0	477.0	477.0	0.0	0.0	0.0	0.0	0.0	2344.0	2344.0	2344.0	0.0
14.0	15/ 1	401.0	468.0	442.9	19.4	5.0	55.0	20.9	15.1	2310.0	2367.0	2330.8	16.3
91.0	15/ 2	337.0	455.0	408.9	26.3	12.0	99.0	41.7	20.5	2276.0	2367.0	2317.6	22.1
364.0	15/ 3	282.0	439.0	374.8	30.9	20.0	132.0	62.6	24.0	2244.0	2367.0	2304.4	25.9
1001.0	15/ 4	241.0	422.0	340.7	34.0	28.0	165.0	83.4	26.4	2212.0	2364.0	2291.1	28.5
2002.0	15/ 5	200.0	405.0	306.6	36.0	37.0	195.0	104.3	28.0	2183.0	2358.0	2277.9	30.2
3003.0	15/ 6	163.0	379.0	272.6	37.2	49.0	216.0	125.1	29.0	2171.0	2350.0	2264.7	31.2
3432.0	15/ 7	130.0	347.0	238.5	37.6	62.0	230.0	146.0	29.3	2162.0	2341.0	2251.5	31.6
3003.0	15/ 8	98.0	314.0	204.4	37.2	76.0	243.0	166.9	29.0	2153.0	2332.0	2238.3	31.2
2002.0	15/ 9	72.0	277.0	170.4	36.0	97.0	255.0	187.7	28.0	2145.0	2320.0	2225.1	30.2
1001.0	15/10	55.0	236.0	136.3	34.0	127.0	264.0	208.6	26.4	2139.0	2291.0	2211.9	28.5
364.0	15/11	38.0	195.0	102.2	30.9	160.0	272.0	229.4	24.0	2136.0	2259.0	2198.6	25.9
91.0	15/12	22.0	140.0	68.1	26.3	193.0	280.0	250.3	20.5	2136.0	2227.0	2185.4	22.1
14.0	15/13	9.0	76.0	34.1	19.4	237.0	287.0	271.1	15.1	2136.0	2193.0	2172.2	16.3
1.0	15/14	0.0	0.0	0.0	0.0	292.0	292.0	292.0	0.0	2159.0	2159.0	2159.0	0.0
Tot:	16384.0	0.0	477.0	238.5	73.3	0.0	292.0	146.0	48.1	2136.0	2367.0	2251.5	39.2

n = 15 14 13 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 1981.00)

1.0	16/ 0	503.0	503.0	503.0	0.0	0.0	0.0	0.0	0.0	2484.0	2484.0	2484.0	0.0
15.0	16/ 1	427.0	494.0	469.5	18.8	5.0	55.0	20.0	15.0	2450.0	2507.0	2470.4	15.8
105.0	16/ 2	363.0	481.0	435.9	25.6	12.0	99.0	39.9	20.4	2416.0	2507.0	2456.8	21.5
455.0	16/ 3	308.0	465.0	402.4	30.2	19.0	132.0	59.8	24.0	2384.0	2507.0	2443.2	25.3
1365.0	16/ 4	267.0	448.0	368.9	33.4	27.0	165.0	79.7	26.5	2352.0	2504.0	2429.6	28.0
3003.0	16/ 5	226.0	431.0	335.3	35.6	35.0	195.0	99.7	28.3	2323.0	2498.0	2416.0	29.8
5005.0	16/ 6	189.0	405.0	301.8	37.0	44.0	216.0	119.6	29.4	2304.0	2490.0	2402.4	31.0
6435.0	16/ 7	156.0	379.0	268.3	37.6	56.0	230.0	139.5	30.0	2292.0	2481.0	2388.8	31.5
6435.0	16/ 8	124.0	347.0	234.7	37.6	69.0	243.0	159.5	30.0	2283.0	2472.0	2375.2	31.5
5005.0	16/ 9	98.0	314.0	201.2	37.0	83.0	255.0	179.4	29.4	2274.0	2460.0	2361.6	31.0
3003.0	16/10	72.0	277.0	167.7	35.6	104.0	264.0	199.3	28.3	2266.0	2441.0	2348.0	29.8
1365.0	16/11	55.0	236.0	134.1	33.4	134.0	272.0	219.3	26.5	2260.0	2412.0	2334.4	28.0
455.0	16/12	38.0	195.0	100.6	30.2	167.0	280.0	239.2	24.0	2257.0	2380.0	2320.8	25.3
105.0	16/13	22.0	140.0	67.1	25.6	200.0	287.0	259.1	20.4	2257.0	2348.0	2307.2	21.5
15.0	16/14	9.0	76.0	33.5	18.8	244.0	294.0	279.1	15.0	2257.0	2314.0	2293.6	15.8
1.0	16/15	0.0	0.0	0.0	0.0	299.0	299.0	299.0	0.0	2280.0	2280.0	2280.0	0.0
Tot:	32768.0	0.0	503.0	251.5	74.5	0.0	299.0	149.5	48.3	2257.0	2507.0	2382.0	40.3

n = 16 15 14 13 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 2081.00)

1.0	17/ 0	535.0	535.0	535.0	0.0	0.0	0.0	0.0	0.0	2616.0	2616.0	2616.0	0.0
16.0	17/ 1	459.0	526.0	501.6	18.2	5.0	55.0	19.0	14.9	2582.0	2639.0	2601.6	15.6
120.0	17/ 2	395.0	513.0	468.1	24.9	10.0	99.0	38.0	20.4	2548.0	2639.0	2587.1	21.3
560.0	17/ 3	340.0	497.0	434.7	29.4	17.0	132.0	57.0	24.1	2516.0	2639.0	2572.7	25.2
1820.0	17/ 4	299.0	480.0	401.3	32.6	24.0	165.0	76.0	26.7	2484.0	2636.0	2558.3	27.9
4368.0	17/ 5	258.0	463.0	367.8	34.9	32.0	195.0	95.0	28.6	2455.0	2630.0	2543.8	29.9
8008.0	17/ 6	221.0	437.0	334.4	36.4	40.0	216.0	114.0	29.9	2428.0	2622.0	2529.4	31.2
11440.0	17/ 7	188.0	411.0	300.9	37.3	49.0	230.0	133.0	30.6	2409.0	2613.0	2514.9	32.0
12870.0	17/ 8	156.0	379.0	267.5	37.6	61.0	243.0	152.0	30.9	2397.0	2604.0	2500.5	32.2
11440.0	17/ 9	124.0	347.0	234.1	37.3	74.0	255.0	171.0	30.6	2388.0	2592.0	2486.1	32.0
8008.0	17/10	98.0	314.0	200.6	36.4	88.0	264.0	190.0	29.9	2379.0	2573.0	2471.6	31.2
4368.0	17/11	72.0	277.0	167.2	34.9	109.0	272.0	209.0	28.6	2371.0	2546.0	2457.2	29.9
1820.0	17/12	55.0	236.0	133.8	32.6	139.0	280.0	228.0	26.7	2365.0	2517.0	2442.8	27.9
560.0	17/13	38.0	195.0	100.3	29.4	172.0	287.0	247.0	24.1	2362.0	2485.0	2428.3	25.2
120.0	17/14	22.0	140.0	66.9	24.9	205.0	294.0	266.0	20.4	2362.0	2453.0	2413.9	21.3
16.0	17/15	9.0	76.0	33.4	18.2	249.0	299.0	285.0	14.9	2362.0	2419.0	2399.4	15.6
1.0	17/16	0.0	0.0	0.0	0.0	304.0	304.0	304.0	0.0	2385.0	2385.0	2385.0	0.0
Tot:	65536.0	0.0	535.0	267.5	76.2	0.0	304.0	152.0	48.3	2362.0	2639.0	2500.5	42.5

n = 17 16 15 14 13 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 2207.00)

1.0	18/ 0	547.0	547.0	547.0	0.0	0.0	0.0	0.0	0.0	2754.0	2754.0	2754.0	0.0
17.0	18/ 1	471.0	538.0	514.8	18.4	3.0	55.0	18.1	15.0	2720.0	2777.0	2739.9	15.2
136.0	18/ 2	407.0	526.0	482.6	25.2	8.0	99.0	36.1	20.5	2686.0	2777.0	2725.8	20.8
680.0	18/ 3	352.0	513.0	450.5	29.8	13.0	132.0	54.2	24.3	2654.0	2777.0	2711.6	24.6
2380.0	18/ 4	311.0	497.0	418.3	33.1	20.0	165.0	72.2	27.0	2622.0	2774.0	2697.5	27.4
6188.0	18/ 5	270.0	480.0	386.1	35.6	27.0	195.0	90.3	29.0	2593.0	2768.0	2683.4	29.4

12376.018/ 6	233.0	463.0	353.9	37.3	35.0	216.0	108.4	30.4	2566.0	2760.0	2669.3	30.9
19448.018/ 7	200.0	437.0	321.8	38.4	43.0	230.0	126.4	31.3	2547.0	2751.0	2655.2	31.8
24310.018/ 8	168.0	411.0	289.6	39.0	52.0	243.0	144.5	31.8	2535.0	2742.0	2641.1	32.2
24310.018/ 9	136.0	379.0	257.4	39.0	64.0	255.0	162.5	31.8	2526.0	2733.0	2626.9	32.2
19448.018/10	110.0	347.0	225.2	38.4	77.0	264.0	180.6	31.3	2517.0	2721.0	2612.8	31.8
12376.018/11	84.0	314.0	193.1	37.3	91.0	272.0	198.6	30.4	2508.0	2702.0	2598.7	30.9
6188.0 18/12	67.0	277.0	160.9	35.6	112.0	280.0	216.7	29.0	2500.0	2675.0	2584.6	29.4
2380.0 18/13	50.0	236.0	128.7	33.1	142.0	287.0	234.8	27.0	2494.0	2646.0	2570.5	27.4
680.0 18/14	34.0	195.0	96.5	29.8	175.0	294.0	252.8	24.3	2491.0	2614.0	2556.4	24.6
136.0 18/15	21.0	140.0	64.4	25.2	208.0	299.0	270.9	20.5	2491.0	2582.0	2542.2	20.8
17.0 18/16	9.0	76.0	32.2	18.4	252.0	304.0	288.9	15.0	2491.0	2548.0	2528.1	15.2
1.0 18/17	0.0	0.0	0.0	0.0	307.0	307.0	307.0	0.0	2514.0	2514.0	2514.0	0.0
Tot: 131072.0	0.0	547.0	273.5	76.4	0.0	307.0	153.5	48.4	2491.0	2777.0	2634.0	42.8

n = 18 17 16 15 14 13 12 11 10 7 6 4 3 2 0 9 8 5 1 (Eval Cost :: 2253.00)

1.0 19/ 0	554.0	554.0	554.0	0.0	0.0	0.0	0.0	0.0	2807.0	2807.0	2807.0	0.0
18.0 19/ 1	478.0	547.0	523.2	18.8	3.0	55.0	17.4	14.8	2773.0	2830.0	2793.7	15.1
153.0 19/ 2	414.0	538.0	492.4	25.8	8.0	99.0	34.9	20.3	2739.0	2830.0	2780.3	20.7
816.0 19/ 3	359.0	526.0	461.7	30.5	13.0	132.0	52.3	24.0	2707.0	2830.0	2767.0	24.6
3060.0 19/ 4	318.0	513.0	430.9	34.1	20.0	165.0	69.8	26.8	2675.0	2830.0	2753.7	27.4
8568.0 19/ 5	277.0	497.0	400.1	36.7	27.0	195.0	87.2	28.9	2646.0	2827.0	2740.3	29.6
18564.019/ 6	240.0	480.0	369.3	38.6	34.0	216.0	104.7	30.4	2619.0	2821.0	2727.0	31.1
31824.019/ 7	207.0	463.0	338.6	40.0	42.0	230.0	122.1	31.4	2600.0	2813.0	2713.7	32.2
43758.019/ 8	175.0	437.0	307.8	40.7	50.0	243.0	139.6	32.0	2588.0	2804.0	2700.3	32.8
48620.019/ 9	143.0	411.0	277.0	41.0	59.0	255.0	157.0	32.2	2579.0	2795.0	2687.0	33.0
43758.019/10	117.0	379.0	246.2	40.7	71.0	264.0	174.4	32.0	2570.0	2786.0	2673.7	32.8
31824.019/11	91.0	347.0	215.4	40.0	84.0	272.0	191.9	31.4	2561.0	2774.0	2660.3	32.2
18564.019/12	74.0	314.0	184.7	38.6	98.0	280.0	209.3	30.4	2553.0	2755.0	2647.0	31.1
8568.0 19/13	57.0	277.0	153.9	36.7	119.0	287.0	226.8	28.9	2547.0	2728.0	2633.7	29.6
3060.0 19/14	41.0	236.0	123.1	34.1	149.0	294.0	244.2	26.8	2544.0	2699.0	2620.3	27.4
816.0 19/15	28.0	195.0	92.3	30.5	182.0	301.0	261.7	24.0	2544.0	2667.0	2607.0	24.6
153.0 19/16	16.0	140.0	61.6	25.8	215.0	306.0	279.1	20.3	2544.0	2635.0	2593.7	20.7
18.0 19/17	7.0	76.0	30.8	18.8	259.0	311.0	296.6	14.8	2544.0	2601.0	2580.3	15.1
1.0 19/18	0.0	0.0	0.0	0.0	314.0	314.0	314.0	0.0	2567.0	2567.0	2567.0	0.0
Tot: 262144.0	0.0	554.0	277.0	76.5	0.0	314.0	157.0	48.5	2544.0	2830.0	2687.0	42.8

****
GRAPH_THREE_DATA_SUMMARY
****

		C(init)				C(test)				C(total)			
Num_Graphs nds/arc	min	max	mean	std_dev	min	max	mean	std_dev	min	max	mean	std_dev
n = 9 1 0 (Eval Cost :: 116.00)												
1.0 3/ 0	42.0	42.0	42.0	0.0	0.0	0.0	0.0	0.0	158.0	158.0	158.0	0.0
2.0 3/ 1	9.0	33.0	21.0	12.0	9.0	33.0	21.0	12.0	158.0	158.0	158.0	0.0
1.0 3/ 2	0.0	0.0	0.0	0.0	42.0	42.0	42.0	0.0	158.0	158.0	158.0	0.0
Tot: 4.0	0.0	42.0	21.0	17.1	0.0	42.0	21.0	17.1	158.0	158.0	158.0	0.0
n = 2 9 1 0 (Eval Cost :: 152.00)												
1.0 4/ 0	74.0	74.0	74.0	0.0	0.0	0.0	0.0	0.0	226.0	226.0	226.0	0.0
3.0 4/ 1	41.0	65.0	49.3	11.1	9.0	55.0	32.3	18.8	226.0	249.0	233.7	10.8
3.0 4/ 2	9.0	33.0	24.7	11.1	42.0	88.0	64.7	18.8	226.0	249.0	241.3	10.8
1.0 4/ 3	0.0	0.0	0.0	0.0	97.0	97.0	97.0	0.0	249.0	249.0	249.0	0.0
Tot: 8.0	0.0	74.0	37.0	23.4	0.0	97.0	48.5	32.4	226.0	249.0	237.5	11.5
n = 3 2 9 1 0 (Eval Cost :: 156.00)												
1.0 5/ 0	91.0	91.0	91.0	0.0	0.0	0.0	0.0	0.0	247.0	247.0	247.0	0.0
4.0 5/ 1	58.0	82.0	68.3	10.2	8.0	55.0	26.3	19.4	238.0	270.0	250.5	11.8
6.0 5/ 2	26.0	65.0	45.5	11.7	17.0	88.0	52.5	22.4	238.0	270.0	254.0	13.7
4.0 5/ 3	9.0	33.0	22.8	10.2	50.0	97.0	78.8	19.4	238.0	270.0	257.5	11.8
1.0 5/ 4	0.0	0.0	0.0	0.0	105.0	105.0	105.0	0.0	261.0	261.0	261.0	0.0
Tot: 16.0	0.0	91.0	45.5	24.9	0.0	105.0	52.5	32.6	238.0	270.0	254.0	12.3
n = 4 3 2 9 1 0 (Eval Cost :: 924.00)												
1.0 6/ 0	107.0	107.0	107.0	0.0	0.0	0.0	0.0	0.0	1031.0	1031.0	1031.0	0.0

5.0	6/ 1	74.0	98.0	85.6	9.5	8.0	55.0	23.6	18.1	1022.0	1054.0	1033.2	10.9
10.0	6/ 2	42.0	82.0	64.2	11.6	17.0	88.0	47.2	22.2	1019.0	1054.0	1035.4	13.4
10.0	6/ 3	25.0	65.0	42.8	11.6	30.0	101.0	70.8	22.2	1019.0	1054.0	1037.6	13.4
5.0	6/ 4	9.0	33.0	21.4	9.5	63.0	110.0	94.4	18.1	1019.0	1051.0	1039.8	10.9
2.0	6/ 5	0.0	0.0	0.0	0.0	118.0	118.0	118.0	0.0	1042.0	1042.0	1042.0	0.0
Tot: 32.0		0.0	107.0	53.5	26.2	0.0	118.0	59.0	33.3	1019.0	1054.0	1036.5	12.4

n = 5 4 3 2 9 1 0 (Eval Cost :: 966.00)

1.0	7/ 0	133.0	133.0	133.0	0.0	0.0	0.0	0.0	0.0	1099.0	1099.0	1099.0	0.0
6.0	7/ 1	100.0	124.0	110.8	8.8	8.0	55.0	22.0	16.9	1087.0	1122.0	1098.8	11.3
15.0	7/ 2	68.0	108.0	88.7	11.2	17.0	88.0	44.0	21.4	1078.0	1122.0	1098.7	14.3
20.0	7/ 3	42.0	91.0	66.5	11.8	30.0	102.0	66.0	22.7	1075.0	1122.0	1098.5	15.1
15.0	7/ 4	25.0	65.0	44.3	11.2	44.0	115.0	88.0	21.4	1075.0	1119.0	1098.3	14.3
6.0	7/ 5	9.0	33.0	22.2	8.8	77.0	124.0	110.0	16.9	1075.0	1110.0	1098.2	11.3
1.0	7/ 6	0.0	0.0	0.0	0.0	132.0	132.0	132.0	0.0	1098.0	1098.0	1098.0	0.0
Tot: 64.0		0.0	133.0	66.5	29.2	0.0	132.0	66.0	34.0	1075.0	1122.0	1098.5	13.8

n = 6 5 4 3 2 9 1 0 (Eval Cost :: 996.00)

1.0	8/ 0	146.0	146.0	146.0	0.0	0.0	0.0	0.0	0.0	1142.0	1142.0	1142.0	0.0
7.0	8/ 1	113.0	137.0	125.1	8.8	7.0	55.0	19.9	16.5	1130.0	1165.0	1141.0	10.6
21.0	8/ 2	81.0	124.0	104.3	11.3	15.0	88.0	39.7	21.3	1121.0	1165.0	1140.0	13.7
35.0	8/ 3	55.0	108.0	83.4	12.4	24.0	102.0	59.6	23.4	1115.0	1165.0	1139.0	15.0
35.0	8/ 4	38.0	91.0	62.6	12.4	37.0	115.0	79.4	23.4	1112.0	1162.0	1138.0	15.0
21.0	8/ 5	22.0	65.0	41.7	11.3	51.0	124.0	99.3	21.3	1112.0	1156.0	1137.0	13.7
7.0	8/ 6	9.0	33.0	20.9	8.8	84.0	132.0	119.1	16.5	1112.0	1147.0	1136.0	10.6
1.0	8/ 7	0.0	0.0	0.0	0.0	139.0	139.0	139.0	0.0	1135.0	1135.0	1135.0	0.0
Tot: 128.0		0.0	146.0	73.0	30.0	0.0	139.0	69.5	34.2	1112.0	1165.0	1138.5	14.1

n = 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1221.00)

1.0	9/ 0	187.0	187.0	187.0	0.0	0.0	0.0	0.0	0.0	1408.0	1408.0	1408.0	0.0
8.0	9/ 1	146.0	178.0	163.6	10.6	7.0	55.0	21.5	16.1	1396.0	1431.0	1406.1	10.2
28.0	9/ 2	113.0	165.0	140.3	13.8	15.0	88.0	43.0	21.0	1387.0	1431.0	1404.3	13.4
56.0	9/ 3	81.0	149.0	116.9	15.5	24.0	121.0	64.5	23.5	1379.0	1431.0	1402.4	15.0
70.0	9/ 4	55.0	132.0	93.5	16.0	37.0	135.0	86.0	24.3	1373.0	1428.0	1400.5	15.4
56.0	9/ 5	38.0	106.0	70.1	15.5	51.0	148.0	107.5	23.5	1370.0	1422.0	1398.6	15.0
28.0	9/ 6	22.0	74.0	46.8	13.8	84.0	157.0	129.0	21.0	1370.0	1414.0	1396.8	13.4
8.0	9/ 7	9.0	41.0	23.4	10.6	117.0	165.0	150.5	16.1	1370.0	1405.0	1394.9	10.2
1.0	9/ 8	0.0	0.0	0.0	0.0	172.0	172.0	172.0	0.0	1393.0	1393.0	1393.0	0.0
Tot: 256.0		0.0	187.0	93.5	36.3	0.0	172.0	86.0	38.0	1370.0	1431.0	1400.5	14.7

n = 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1261.00)

1.0	10/ 0	204.0	204.0	204.0	0.0	0.0	0.0	0.0	0.0	1465.0	1465.0	1465.0	0.0
9.0	10/ 1	163.0	195.0	181.3	10.2	7.0	55.0	20.0	15.7	1453.0	1488.0	1462.3	9.9
36.0	10/ 2	130.0	182.0	158.7	13.4	15.0	88.0	40.0	20.8	1444.0	1488.0	1459.7	13.1
84.0	10/ 3	98.0	166.0	136.0	15.2	23.0	121.0	60.0	23.6	1435.0	1488.0	1457.0	14.8
126.0	10/ 4	72.0	149.0	113.3	16.1	32.0	135.0	80.0	24.9	1427.0	1485.0	1454.3	15.6
126.0	10/ 5	55.0	132.0	90.7	16.1	45.0	148.0	100.0	24.9	1421.0	1479.0	1451.7	15.6
84.0	10/ 6	38.0	106.0	68.0	15.2	59.0	157.0	120.0	23.6	1418.0	1471.0	1449.0	14.8
36.0	10/ 7	22.0	74.0	45.3	13.4	92.0	165.0	140.0	20.8	1418.0	1462.0	1446.3	13.1
9.0	10/ 8	9.0	41.0	22.7	10.2	125.0	173.0	160.0	15.7	1418.0	1453.0	1443.7	9.9
1.0	10/ 9	0.0	0.0	0.0	0.0	180.0	180.0	180.0	0.0	1441.0	1441.0	1441.0	0.0
Tot: 512.0		0.0	204.0	102.0	37.3	0.0	180.0	90.0	38.2	1418.0	1488.0	1453.0	15.4

n = 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1362.00)

1.0	11/ 0	259.0	259.0	259.0	0.0	0.0	0.0	0.0	0.0	1621.0	1621.0	1621.0	0.0
10.0	11/ 1	204.0	250.0	233.1	13.7	7.0	55.0	20.1	14.9	1587.0	1644.0	1615.2	13.3
45.0	11/ 2	163.0	237.0	207.2	18.2	15.0	88.0	40.2	19.9	1575.0	1644.0	1609.4	17.7
120.0	11/ 3	130.0	221.0	181.3	20.9	23.0	121.0	60.3	22.8	1566.0	1644.0	1603.6	20.3
210.0	11/ 4	98.0	204.0	155.4	22.3	32.0	142.0	80.4	24.4	1557.0	1641.0	1597.8	21.7
252.0	11/ 5	72.0	187.0	129.5	22.8	45.0	156.0	100.5	24.9	1549.0	1635.0	1592.0	22.1
210.0	11/ 6	55.0	161.0	103.6	22.3	59.0	169.0	120.6	24.4	1543.0	1627.0	1586.2	21.7
120.0	11/ 7	38.0	129.0	77.7	20.9	80.0	178.0	140.7	22.8	1540.0	1618.0	1580.4	20.3
45.0	11/ 8	22.0	96.0	51.8	18.2	113.0	186.0	160.8	19.9	1540.0	1609.0	1574.6	17.7
10.0	11/ 9	9.0	55.0	25.9	13.7	146.0	194.0	180.9	14.9	1540.0	1597.0	1568.8	13.3
1.0	11/10	0.0	0.0	0.0	0.0	201.0	201.0	201.0	0.0	1563.0	1563.0	1563.0	0.0

Tot:	1024.0		0.0	259.0	129.5	46.3	0.0	201.0	100.5	39.6	1540.0	1644.0	1592.0	22.9

n = 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1382.00)

1.0	12/ 0	300.0	300.0	300.0	0.0	0.0	0.0	0.0	0.0	1682.0	1682.0	1682.0	0.0
11.0	12/ 1	245.0	291.0	272.7	13.7	7.0	55.0	19.4	14.4	1648.0	1705.0	1674.1	14.3
55.0	12/ 2	204.0	278.0	245.5	18.4	15.0	98.0	38.7	19.3	1619.0	1705.0	1666.2	19.2
165.0	12/ 3	163.0	262.0	218.2	21.3	23.0	121.0	58.1	22.3	1607.0	1705.0	1658.3	22.2
330.0	12/ 4	130.0	245.0	190.9	23.0	32.0	142.0	77.5	24.1	1598.0	1702.0	1650.4	24.0
462.0	12/ 5	98.0	228.0	163.6	23.8	44.0	156.0	96.8	25.0	1589.0	1696.0	1642.5	24.8
462.0	12/ 6	72.0	202.0	136.4	23.8	57.0	169.0	116.2	25.0	1581.0	1688.0	1634.5	24.8
330.0	12/ 7	55.0	170.0	109.1	23.0	71.0	181.0	135.5	24.1	1575.0	1679.0	1626.6	24.0
165.0	12/ 8	38.0	137.0	81.8	21.3	92.0	190.0	154.9	22.3	1572.0	1670.0	1618.7	22.2
55.0	12/ 9	22.0	96.0	54.5	18.4	125.0	198.0	174.3	19.3	1572.0	1658.0	1610.8	19.2
11.0	12/10	9.0	55.0	27.3	13.7	158.0	206.0	193.6	14.4	1572.0	1629.0	1602.9	14.3
1.0	12/11	0.0	0.0	0.0	0.0	213.0	213.0	213.0	0.0	1595.0	1595.0	1595.0	0.0
Tot: 2048.0		0.0	300.0	150.0	50.6	0.0	213.0	106.5	40.0	1572.0	1705.0	1638.5	27.1

n = 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1603.00)

1.0	13/ 0	376.0	376.0	376.0	0.0	0.0	0.0	0.0	0.0	1979.0	1979.0	1979.0	0.0
12.0	13/ 1	300.0	367.0	344.7	18.8	7.0	55.0	21.4	15.4	1945.0	2002.0	1969.1	15.2
66.0	13/ 2	245.0	354.0	313.3	25.4	15.0	99.0	42.8	20.8	1913.0	2002.0	1959.2	20.5
220.0	13/ 3	204.0	338.0	282.0	29.5	23.0	132.0	64.3	24.1	1884.0	2002.0	1949.3	23.9
495.0	13/ 4	163.0	321.0	250.7	32.1	32.0	165.0	85.7	26.3	1872.0	2000.0	1939.3	26.0
792.0	13/ 5	130.0	304.0	219.3	33.6	44.0	186.0	107.1	27.5	1863.0	1993.0	1929.4	27.2
924.0	13/ 6	98.0	278.0	188.0	34.1	57.0	200.0	128.5	27.8	1854.0	1985.0	1919.5	27.6
792.0	13/ 7	72.0	246.0	156.7	33.6	71.0	213.0	150.0	27.5	1846.0	1976.0	1909.6	27.2
495.0	13/ 8	55.0	213.0	125.3	32.1	92.0	225.0	171.3	26.3	1840.0	1967.0	1900.0	26.0
220.0	13/ 9	38.0	172.0	94.0	29.5	125.0	234.0	192.8	24.1	1837.0	1955.0	1889.8	23.9
66.0	13/10	22.0	131.0	62.7	25.4	158.0	242.0	214.2	20.8	1837.0	1926.0	1879.8	20.5
12.0	13/11	9.0	76.0	31.3	18.8	202.0	250.0	235.6	15.4	1837.0	1894.0	1870.0	15.2
1.0	13/12	0.0	0.0	0.0	0.0	257.0	257.0	257.0	0.0	1860.0	1860.0	1860.0	0.0
Tot: 4096.0		0.0	376.0	188.0	63.3	0.0	257.0	128.5	45.7	1837.0	2002.0	1919.5	31.5

n = 13 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1657.00)

1.0	14/ 0	440.0	440.0	440.0	0.0	0.0	0.0	0.0	0.0	2097.0	2097.0	2097.0	0.0
13.0	14/ 1	364.0	431.0	406.2	20.1	7.0	55.0	22.1	15.0	2063.0	2120.0	2085.2	16.0
78.0	14/ 2	300.0	418.0	372.3	27.2	15.0	99.0	44.2	20.3	2029.0	2120.0	2073.5	21.6
286.0	14/ 3	245.0	402.0	338.5	31.7	23.0	132.0	66.2	23.7	2000.0	2120.0	2061.7	25.3
715.0	14/ 4	204.0	385.0	304.6	34.8	32.0	165.0	88.3	25.9	1968.0	2117.0	2050.0	27.7
1287.0	14/ 5	163.0	368.0	270.8	36.7	44.0	195.0	110.4	27.3	1956.0	2111.0	2038.2	29.2
1716.0	14/ 6	130.0	342.0	236.9	37.6	57.0	216.0	132.5	28.0	1947.0	2103.0	2026.4	29.9
1716.0	14/ 7	98.0	310.0	203.1	37.6	71.0	230.0	154.5	28.0	1938.0	2094.0	2014.6	29.9
1287.0	14/ 8	72.0	277.0	169.2	36.7	92.0	243.0	176.6	27.3	1930.0	2085.0	2002.8	29.2
715.0	14/ 9	55.0	236.0	135.4	34.8	122.0	255.0	198.7	25.9	1924.0	2073.0	1991.1	27.7
286.0	14/10	38.0	195.0	101.5	31.7	155.0	264.0	220.8	23.7	1921.0	2044.0	1979.3	25.3
78.0	14/11	22.0	140.0	67.7	27.2	188.0	272.0	242.8	20.3	1921.0	2012.0	1967.5	21.6
13.0	14/12	9.0	76.0	33.8	20.1	232.0	280.0	264.9	15.0	1921.0	1978.0	1955.8	16.0
1.0	14/13	0.0	0.0	0.0	0.0	287.0	287.0	287.0	0.0	1944.0	1944.0	1944.0	0.0
Tot: 8192.0		0.0	440.0	220.0	70.9	0.0	287.0	143.5	48.1	1921.0	2120.0	2020.5	35.8

n = 14 13 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1867.00)

1.0	15/ 0	477.0	477.0	477.0	0.0	0.0	0.0	0.0	0.0	2344.0	2344.0	2344.0	0.0
14.0	15/ 1	401.0	468.0	442.9	19.4	5.0	55.0	20.9	15.1	2310.0	2367.0	2330.8	16.3
91.0	15/ 2	337.0	455.0	408.9	26.3	12.0	99.0	41.7	20.5	2276.0	2367.0	2317.6	22.1
364.0	15/ 3	282.0	439.0	374.8	30.9	20.0	132.0	62.6	24.0	2244.0	2367.0	2304.4	25.9
1001.0	15/ 4	241.0	422.0	340.7	34.0	28.0	165.0	83.4	26.4	2212.0	2364.0	2291.1	28.5
2002.0	15/ 5	200.0	405.0	306.6	36.0	37.0	195.0	104.3	28.0	2183.0	2358.0	2277.9	30.2
3003.0	15/ 6	163.0	379.0	272.6	37.2	49.0	216.0	125.1	29.0	2171.0	2350.0	2264.7	31.2
3432.0	15/ 7	130.0	347.0	238.5	37.6	62.0	230.0	146.0	29.3	2162.0	2341.0	2251.5	31.6
3003.0	15/ 8	98.0	314.0	204.4	37.2	76.0	243.0	166.9	29.0	2153.0	2332.0	2238.3	31.2
2002.0	15/ 9	72.0	277.0	170.4	36.0	97.0	255.0	187.7	28.0	2145.0	2320.0	2225.1	30.2
1001.0	15/10	55.0	236.0	136.3	34.0	127.0	264.0	208.6	26.4	2139.0	2291.0	2211.9	28.5
364.0	15/11	38.0	195.0	102.2	30.9	160.0	272.0	229.4	24.0	2136.0	2259.0	2198.6	25.9
91.0	15/12	22.0	140.0	68.1	26.3	193.0	280.0	250.3	20.5	2136.0	2227.0	2185.4	22.1
14.0	15/13	9.0	76.0	34.1	19.4	237.0	287.0	271.1	15.1	2136.0	2193.0	2172.2	16.3
1.0	15/14	0.0	0.0	0.0	0.0	292.0	292.0	292.0	0.0	2159.0	2159.0	2159.0	0.0

Tot:	16384.0	0.0	477.0	238.5	73.3	0.0	292.0	146.0	48.1	2136.0	2367.0	2251.5	39.2

n = 15 14 13 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1981.00)

1.0	16/ 0	503.0	503.0	503.0	0.0	0.0	0.0	0.0	0.0	2484.0	2484.0	2484.0	0.0
15.0	16/ 1	427.0	494.0	469.5	18.8	5.0	55.0	20.0	15.0	2450.0	2507.0	2470.4	15.8
105.0	16/ 2	363.0	481.0	435.9	25.6	12.0	99.0	39.9	20.4	2416.0	2507.0	2456.8	21.5
455.0	16/ 3	308.0	465.0	402.4	30.2	19.0	132.0	59.8	24.0	2384.0	2507.0	2443.2	25.3
1365.0	16/ 4	267.0	448.0	368.9	33.4	27.0	165.0	79.7	26.5	2352.0	2504.0	2429.6	28.0
3003.0	16/ 5	226.0	431.0	335.3	35.6	35.0	195.0	99.7	28.3	2323.0	2498.0	2416.0	29.8
5005.0	16/ 6	189.0	405.0	301.8	37.0	44.0	216.0	119.6	29.4	2304.0	2490.0	2402.4	31.0
6435.0	16/ 7	156.0	379.0	268.3	37.6	56.0	230.0	139.5	30.0	2292.0	2481.0	2388.8	31.5
6435.0	16/ 8	124.0	347.0	234.7	37.6	69.0	243.0	159.5	30.0	2283.0	2472.0	2375.2	31.5
5005.0	16/ 9	98.0	314.0	201.2	37.0	83.0	255.0	179.4	29.4	2274.0	2460.0	2361.6	31.0
3003.0	16/10	72.0	277.0	167.7	35.6	104.0	264.0	199.3	28.3	2266.0	2441.0	2348.0	29.8
1365.0	16/11	55.0	236.0	134.1	33.4	134.0	272.0	219.3	26.5	2260.0	2412.0	2334.4	28.0
455.0	16/12	38.0	195.0	100.6	30.2	167.0	280.0	239.2	24.0	2257.0	2380.0	2320.8	25.3
105.0	16/13	22.0	140.0	67.1	25.6	200.0	287.0	259.1	20.4	2257.0	2348.0	2307.2	21.5
15.0	16/14	9.0	76.0	33.5	18.8	244.0	294.0	279.1	15.0	2257.0	2314.0	2293.6	15.8
1.0	16/15	0.0	0.0	0.0	0.0	299.0	299.0	299.0	0.0	2280.0	2280.0	2280.0	0.0
Tot: 32768.0		0.0	503.0	251.5	74.5	0.0	299.0	149.5	48.3	2257.0	2507.0	2382.0	40.3

n = 16 15 14 13 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2081.00)

1.0	17/ 0	535.0	535.0	535.0	0.0	0.0	0.0	0.0	0.0	2616.0	2616.0	2616.0	0.0
16.0	17/ 1	459.0	526.0	501.6	18.2	5.0	55.0	19.0	14.9	2582.0	2639.0	2601.6	15.6
120.0	17/ 2	395.0	513.0	468.1	24.9	10.0	99.0	38.0	20.4	2548.0	2639.0	2587.1	21.3
560.0	17/ 3	340.0	497.0	434.7	29.4	17.0	132.0	57.0	24.1	2516.0	2639.0	2572.7	25.2
1820.0	17/ 4	299.0	480.0	401.3	32.6	24.0	165.0	76.0	26.7	2484.0	2636.0	2558.3	27.9
4368.0	17/ 5	258.0	463.0	367.8	34.9	32.0	195.0	95.0	28.6	2455.0	2630.0	2543.8	29.9
8008.0	17/ 6	221.0	437.0	334.4	36.4	40.0	216.0	114.0	29.9	2428.0	2622.0	2529.4	31.2
11440.0	17/ 7	188.0	411.0	300.9	37.3	49.0	230.0	133.0	30.6	2409.0	2613.0	2514.9	32.0
12870.0	17/ 8	156.0	379.0	267.5	37.6	61.0	243.0	152.0	30.9	2397.0	2604.0	2500.5	32.2
11440.0	17/ 9	124.0	347.0	234.1	37.3	74.0	255.0	171.0	30.6	2388.0	2592.0	2486.1	32.0
8008.0	17/10	98.0	314.0	200.6	36.4	88.0	264.0	190.0	29.9	2379.0	2573.0	2471.6	31.2
4368.0	17/11	72.0	277.0	167.2	34.9	109.0	272.0	209.0	28.6	2371.0	2546.0	2457.2	29.9
1820.0	17/12	55.0	236.0	133.8	32.6	139.0	280.0	228.0	26.7	2365.0	2517.0	2442.8	27.9
560.0	17/13	38.0	195.0	100.3	29.4	172.0	287.0	247.0	24.1	2362.0	2485.0	2428.3	25.2
120.0	17/14	22.0	140.0	66.9	24.9	205.0	294.0	266.0	20.4	2362.0	2453.0	2413.9	21.3
16.0	17/15	9.0	76.0	33.4	18.2	249.0	299.0	285.0	14.9	2362.0	2419.0	2399.4	15.6
1.0	17/16	0.0	0.0	0.0	0.0	304.0	304.0	304.0	0.0	2385.0	2385.0	2385.0	0.0
Tot: 65536.0		0.0	535.0	267.5	76.2	0.0	304.0	152.0	48.3	2362.0	2639.0	2500.5	42.5

n = 17 16 15 14 13 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2207.00)

1.0	18/ 0	547.0	547.0	547.0	0.0	0.0	0.0	0.0	0.0	2754.0	2754.0	2754.0	0.0
17.0	18/ 1	471.0	538.0	514.8	18.4	3.0	55.0	18.1	15.0	2720.0	2777.0	2739.9	15.2
136.0	18/ 2	407.0	526.0	482.6	25.2	8.0	99.0	36.1	20.5	2686.0	2777.0	2725.8	20.8
680.0	18/ 3	352.0	513.0	450.5	29.8	13.0	132.0	54.2	24.3	2654.0	2777.0	2711.6	24.6
2380.0	18/ 4	311.0	497.0	418.3	33.1	20.0	165.0	72.2	27.0	2622.0	2774.0	2697.5	27.4
6188.0	18/ 5	270.0	480.0	386.1	35.6	27.0	195.0	90.3	29.0	2593.0	2768.0	2683.4	29.4
12376.0	18/ 6	233.0	463.0	353.9	37.3	35.0	216.0	108.4	30.4	2566.0	2760.0	2669.3	30.9
19448.0	18/ 7	200.0	437.0	321.8	38.4	43.0	230.0	126.4	31.3	2547.0	2751.0	2655.2	31.8
24310.0	18/ 8	168.0	411.0	289.6	39.0	52.0	243.0	144.5	31.8	2535.0	2742.0	2641.1	32.2
24310.0	18/ 9	136.0	379.0	257.4	39.0	64.0	255.0	162.5	31.8	2526.0	2733.0	2626.9	32.2
19448.0	18/10	110.0	347.0	225.2	38.4	77.0	264.0	180.6	31.3	2517.0	2721.0	2612.8	31.8
12376.0	18/11	84.0	314.0	193.1	37.3	91.0	272.0	198.6	30.4	2508.0	2702.0	2598.7	30.9
6188.0	18/12	67.0	277.0	160.9	35.6	112.0	280.0	216.7	29.0	2500.0	2675.0	2584.6	29.4
2380.0	18/13	50.0	236.0	128.7	33.1	142.0	287.0	234.8	27.0	2494.0	2646.0	2570.5	27.4
680.0	18/14	34.0	195.0	96.5	29.8	175.0	294.0	252.8	24.3	2491.0	2614.0	2556.4	24.6
136.0	18/15	21.0	140.0	64.4	25.2	208.0	299.0	270.9	20.5	2491.0	2582.0	2542.2	20.8
17.0	18/16	9.0	76.0	32.2	18.4	252.0	304.0	288.9	15.0	2491.0	2548.0	2528.1	15.2
1.0	18/17	0.0	0.0	0.0	0.0	307.0	307.0	307.0	0.0	2514.0	2514.0	2514.0	0.0
Tot: 131072.0		0.0	547.0	273.5	76.4	0.0	307.0	153.5	48.4	2491.0	2777.0	2634.0	42.8

n = 18 17 16 15 14 13 12 11 10 8 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2253.00)

1.0	19/ 0	554.0	554.0	554.0	0.0	0.0	0.0	0.0	0.0	2807.0	2807.0	2807.0	0.0
18.0	19/ 1	478.0	547.0	523.2	18.8	3.0	55.0	17.4	14.8	2773.0	2830.0	2793.7	15.1
153.0	19/ 2	414.0	538.0	492.4	25.8	8.0	99.0	34.9	20.3	2739.0	2830.0	2780.3	20.7

816.0	19/ 3	359.0	526.0	461.7	30.5	13.0	132.0	52.3	24.0	2707.0	2830.0	2767.0	24.6
3060.0	19/ 4	318.0	513.0	430.9	34.1	20.0	165.0	69.8	26.8	2675.0	2830.0	2753.7	27.4
8568.0	19/ 5	277.0	497.0	400.1	36.7	27.0	195.0	87.2	28.9	2646.0	2827.0	2740.3	29.6
18564.0	19/ 6	240.0	480.0	369.3	38.6	34.0	216.0	104.7	30.4	2619.0	2821.0	2727.0	31.1
31824.0	19/ 7	207.0	463.0	338.6	40.0	42.0	230.0	122.1	31.4	2600.0	2813.0	2713.7	32.2
43758.0	19/ 8	175.0	437.0	307.8	40.7	50.0	243.0	139.6	32.0	2588.0	2804.0	2700.3	32.8
48620.0	19/ 9	143.0	411.0	277.0	41.0	59.0	255.0	157.0	32.2	2579.0	2795.0	2687.0	33.0
43758.0	19/10	117.0	379.0	246.2	40.7	71.0	264.0	174.4	32.0	2570.0	2786.0	2673.7	32.8
31824.0	19/11	91.0	347.0	215.4	40.0	84.0	272.0	191.9	31.4	2561.0	2774.0	2660.3	32.2
18564.0	19/12	74.0	314.0	184.7	38.6	98.0	280.0	209.3	30.4	2553.0	2755.0	2647.0	31.1
8568.0	19/13	57.0	277.0	153.9	36.7	119.0	287.0	226.8	28.9	2547.0	2728.0	2633.7	29.6
3060.0	19/14	41.0	236.0	123.1	34.1	149.0	294.0	244.2	26.8	2544.0	2699.0	2620.3	27.4
816.0	19/15	28.0	195.0	92.3	30.5	182.0	301.0	261.7	24.0	2544.0	2667.0	2607.0	24.6
153.0	19/16	16.0	140.0	61.6	25.8	215.0	306.0	279.1	20.3	2544.0	2635.0	2593.7	20.7
18.0	19/17	7.0	76.0	30.8	18.8	259.0	311.0	296.6	14.8	2544.0	2601.0	2580.3	15.1
1.0	19/18	0.0	0.0	0.0	0.0	314.0	314.0	314.0	0.0	2567.0	2567.0	2567.0	0.0
Tot: 262144.0		0.0	554.0	277.0	76.5	0.0	314.0	157.0	48.5	2544.0	2830.0	2687.0	42.8

****
GRAPH_FOUR_DATA_SUMMARY
****

		C(init)				C(test)				C(total)			
Num_Graphs	nds/arc	min	max	mean	std_dev	min	max	mean	std_dev	min	max	mean	std_dev
n = 9 1 0 (Eval Cost :: 116.00)													
1.0	3/ 0	42.0	42.0	42.0	0.0	0.0	0.0	0.0	0.0	158.0	158.0	158.0	0.0
2.0	3/ 1	9.0	33.0	21.0	12.0	9.0	33.0	21.0	12.0	158.0	158.0	158.0	0.0
1.0	3/ 2	0.0	0.0	0.0	0.0	42.0	42.0	42.0	0.0	158.0	158.0	158.0	0.0
Tot: 4.0		0.0	42.0	21.0	17.1	0.0	42.0	21.0	17.1	158.0	158.0	158.0	0.0
n = 2 9 1 0 (Eval Cost :: 152.00)													
1.0	4/ 0	74.0	74.0	74.0	0.0	0.0	0.0	0.0	0.0	226.0	226.0	226.0	0.0
3.0	4/ 1	41.0	65.0	49.3	11.1	9.0	55.0	32.3	18.8	226.0	249.0	233.7	10.8
3.0	4/ 2	9.0	33.0	24.7	11.1	42.0	88.0	64.7	18.8	226.0	249.0	241.3	10.8
1.0	4/ 3	0.0	0.0	0.0	0.0	97.0	97.0	97.0	0.0	249.0	249.0	249.0	0.0
Tot: 8.0		0.0	74.0	37.0	23.4	0.0	97.0	48.5	32.4	226.0	249.0	237.5	11.5
n = 3 2 9 1 0 (Eval Cost :: 156.00)													
1.0	5/ 0	91.0	91.0	91.0	0.0	0.0	0.0	0.0	0.0	247.0	247.0	247.0	0.0
4.0	5/ 1	58.0	82.0	68.3	10.2	8.0	55.0	26.3	19.4	238.0	270.0	250.5	11.8
6.0	5/ 2	26.0	65.0	45.5	11.7	17.0	88.0	52.5	22.4	238.0	270.0	254.0	13.7
4.0	5/ 3	9.0	33.0	22.8	10.2	50.0	97.0	78.8	19.4	238.0	270.0	257.5	11.8
1.0	5/ 4	0.0	0.0	0.0	0.0	105.0	105.0	105.0	0.0	261.0	261.0	261.0	0.0
Tot: 16.0		0.0	91.0	45.5	24.9	0.0	105.0	52.5	32.6	238.0	270.0	254.0	12.3
n = 4 3 2 9 1 0 (Eval Cost :: 924.00)													
1.0	6/ 0	107.0	107.0	107.0	0.0	0.0	0.0	0.0	0.0	1031.0	1031.0	1031.0	0.0
5.0	6/ 1	74.0	98.0	85.6	9.5	8.0	55.0	23.6	18.1	1022.0	1054.0	1033.2	10.9
10.0	6/ 2	42.0	82.0	64.2	11.6	17.0	88.0	47.2	22.2	1019.0	1054.0	1035.4	13.4
10.0	6/ 3	25.0	65.0	42.8	11.6	30.0	101.0	70.8	22.2	1019.0	1054.0	1037.6	13.4
5.0	6/ 4	9.0	33.0	21.4	9.5	63.0	110.0	94.4	18.1	1019.0	1051.0	1039.8	10.9
1.0	6/ 5	0.0	0.0	0.0	0.0	118.0	118.0	118.0	0.0	1042.0	1042.0	1042.0	0.0
Tot: 32.0		0.0	107.0	53.5	26.2	0.0	118.0	59.0	33.3	1019.0	1054.0	1036.5	12.4
n = 5 4 3 2 9 1 0 (Eval Cost :: 966.00)													
1.0	7/ 0	133.0	133.0	133.0	0.0	0.0	0.0	0.0	0.0	1099.0	1099.0	1099.0	0.0
6.0	7/ 1	100.0	124.0	110.8	8.8	8.0	55.0	22.0	16.9	1087.0	1122.0	1098.8	11.3
15.0	7/ 2	68.0	108.0	88.7	11.2	17.0	88.0	44.0	21.4	1078.0	1122.0	1098.7	14.3
20.0	7/ 3	42.0	91.0	66.5	11.8	30.0	102.0	66.0	22.7	1075.0	1122.0	1098.5	15.1
15.0	7/ 4	25.0	65.0	44.3	11.2	44.0	115.0	88.0	21.4	1075.0	1119.0	1098.3	14.3
6.0	7/ 5	9.0	33.0	22.2	8.8	77.0	124.0	110.0	16.9	1075.0	1110.0	1098.2	11.3

	1.0	7/ 6	0.0	0.0	0.0	0.0	132.0	132.0	132.0	0.0	1098.0	1098.0	1098.0	0.0
Tot:	64.0		0.0	133.0	66.5	29.2	0.0	132.0	66.0	34.0	1075.0	1122.0	1098.5	13.8

n = 6 5 4 3 2 9 1 0 (Eval Cost :: 996.00)

1.0	8/ 0	146.0	146.0	146.0	0.0	0.0	0.0	0.0	0.0	1142.0	1142.0	1142.0	0.0
7.0	8/ 1	113.0	137.0	125.1	8.8	7.0	55.0	19.9	16.5	1130.0	1165.0	1141.0	10.6
21.0	8/ 2	81.0	124.0	104.3	11.3	15.0	88.0	39.7	21.3	1121.0	1165.0	1140.0	13.7
35.0	8/ 3	55.0	108.0	83.4	12.4	24.0	102.0	59.6	23.4	1115.0	1165.0	1139.0	15.0
35.0	8/ 4	38.0	91.0	62.6	12.4	37.0	115.0	79.4	23.4	1112.0	1162.0	1138.0	15.0
21.0	8/ 5	22.0	65.0	41.7	11.3	51.0	124.0	99.3	21.3	1112.0	1156.0	1137.0	13.7
7.0	8/ 6	9.0	33.0	20.9	8.8	84.0	132.0	119.1	16.5	1112.0	1147.0	1136.0	10.6
1.0	8/ 7	0.0	0.0	0.0	0.0	139.0	139.0	139.0	0.0	1135.0	1135.0	1135.0	0.0
Tot: 128.0		0.0	146.0	73.0	30.0	0.0	139.0	69.5	34.2	1112.0	1165.0	1138.5	14.1

n = 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1221.00)

1.0	9/ 0	187.0	187.0	187.0	0.0	0.0	0.0	0.0	0.0	1408.0	1408.0	1408.0	0.0
8.0	9/ 1	146.0	178.0	163.6	10.6	7.0	55.0	21.5	16.1	1396.0	1431.0	1406.1	10.2
28.0	9/ 2	113.0	165.0	140.3	13.8	15.0	88.0	43.0	21.0	1387.0	1431.0	1404.3	13.4
56.0	9/ 3	81.0	149.0	116.9	15.5	24.0	121.0	64.5	23.5	1379.0	1431.0	1402.4	15.0
70.0	9/ 4	55.0	132.0	93.5	16.0	37.0	135.0	86.0	24.3	1373.0	1428.0	1400.5	15.4
56.0	9/ 5	38.0	106.0	70.1	15.5	51.0	148.0	107.5	23.5	1370.0	1422.0	1398.6	15.0
28.0	9/ 6	22.0	74.0	46.8	13.8	84.0	157.0	129.0	21.0	1370.0	1414.0	1396.8	13.4
8.0	9/ 7	9.0	41.0	23.4	10.6	117.0	165.0	150.5	16.1	1370.0	1405.0	1394.9	10.2
1.0	9/ 8	0.0	0.0	0.0	0.0	172.0	172.0	172.0	0.0	1393.0	1393.0	1393.0	0.0
Tot: 256.0		0.0	187.0	93.5	36.3	0.0	172.0	86.0	38.0	1370.0	1431.0	1400.5	14.7

n = 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1322.00)

1.0	10/ 0	242.0	242.0	242.0	0.0	0.0	0.0	0.0	0.0	1564.0	1564.0	1564.0	0.0
9.0	10/ 1	187.0	233.0	215.1	14.1	7.0	55.0	21.4	15.1	1530.0	1587.0	1558.6	14.0
36.0	10/ 2	146.0	220.0	188.2	18.6	15.0	88.0	42.9	20.0	1518.0	1587.0	1553.1	18.5
84.0	10/ 3	113.0	204.0	161.3	21.1	24.0	121.0	64.3	22.7	1509.0	1587.0	1547.7	20.9
126.0	10/ 4	81.0	187.0	134.4	22.3	37.0	142.0	85.8	24.0	1501.0	1584.0	1542.2	22.1
126.0	10/ 5	55.0	161.0	107.6	22.3	51.0	156.0	107.2	24.0	1495.0	1578.0	1536.8	22.1
84.0	10/ 6	38.0	129.0	80.7	21.1	72.0	169.0	128.7	22.7	1492.0	1570.0	1531.3	20.9
36.0	10/ 7	22.0	96.0	53.8	18.6	105.0	178.0	150.1	20.0	1492.0	1561.0	1525.9	18.5
9.0	10/ 8	9.0	55.0	26.9	14.1	138.0	186.0	171.6	15.1	1492.0	1549.0	1520.4	14.0
1.0	10/ 9	0.0	0.0	0.0	0.0	193.0	193.0	193.0	0.0	1515.0	1515.0	1515.0	0.0
Tot: 512.0		0.0	242.0	121.0	45.5	0.0	193.0	96.5	39.4	1492.0	1587.0	1539.5	22.5

n = 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1342.00)

1.0	11/ 0	283.0	283.0	283.0	0.0	0.0	0.0	0.0	0.0	1625.0	1625.0	1625.0	0.0
10.0	11/ 1	228.0	274.0	254.7	14.0	7.0	55.0	20.5	14.6	1591.0	1648.0	1617.2	15.0
45.0	11/ 2	187.0	261.0	226.4	18.7	15.0	88.0	41.0	19.5	1562.0	1648.0	1609.4	20.0
120.0	11/ 3	146.0	245.0	198.1	21.4	24.0	121.0	61.5	22.4	1550.0	1648.0	1601.6	22.9
210.0	11/ 4	113.0	228.0	169.8	22.9	36.0	142.0	82.0	23.9	1541.0	1645.0	1593.8	24.5
252.0	11/ 5	81.0	202.0	141.5	23.3	49.0	156.0	102.5	24.4	1533.0	1639.0	1586.0	25.0
210.0	11/ 6	55.0	170.0	113.2	22.9	63.0	169.0	123.0	23.9	1527.0	1631.0	1578.2	24.5
120.0	11/ 7	38.0	137.0	84.9	21.4	84.0	181.0	143.5	22.4	1524.0	1622.0	1570.4	22.9
45.0	11/ 8	22.0	96.0	56.6	18.7	117.0	190.0	164.0	19.5	1524.0	1610.0	1562.6	20.0
10.0	11/ 9	9.0	55.0	28.3	14.0	150.0	198.0	184.5	14.6	1524.0	1581.0	1554.8	15.0
1.0	11/10	0.0	0.0	0.0	0.0	205.0	205.0	205.0	0.0	1547.0	1547.0	1547.0	0.0
Tot: 1024.0		0.0	283.0	141.5	50.0	0.0	205.0	102.5	39.8	1524.0	1648.0	1586.0	26.7

n = 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1563.00)

1.0	12/ 0	359.0	359.0	359.0	0.0	0.0	0.0	0.0	0.0	1922.0	1922.0	1922.0	0.0
11.0	12/ 1	283.0	350.0	326.4	19.1	7.0	55.0	22.6	15.5	1888.0	1945.0	1912.0	15.9
55.0	12/ 2	228.0	337.0	293.7	25.7	15.0	99.0	45.3	20.8	1856.0	1945.0	1902.0	21.3
165.0	12/ 3	187.0	321.0	261.1	29.7	24.0	132.0	67.9	24.0	1827.0	1945.0	1892.0	24.6
330.0	12/ 4	146.0	304.0	228.5	32.0	36.0	165.0	90.5	26.0	1815.0	1942.0	1882.0	26.6
462.0	12/ 5	113.0	278.0	195.8	33.2	49.0	186.0	113.2	26.9	1806.0	1936.0	1872.0	27.6

	462.0	12/ 6	81.0	246.0	163.2	33.2	63.0	200.0	135.8	26.9	1798.0	1928.0	1862.0	27.6
	330.0	12/ 7	55.0	213.0	130.5	32.0	84.0	213.0	158.5	26.0	1792.0	1919.0	1852.0	26.6
	165.0	12/ 8	38.0	172.0	97.9	29.7	117.0	225.0	181.1	24.0	1789.0	1907.0	1842.0	24.6
	55.0	12/ 9	22.0	131.0	65.3	25.7	150.0	234.0	203.7	20.8	1789.0	1878.0	1832.0	21.3
	11.0	12/10	9.0	76.0	32.6	19.1	194.0	242.0	226.4	15.5	1789.0	1846.0	1822.0	15.9
	1.0	12/11	0.0	0.0	0.0	0.0	249.0	249.0	249.0	0.0	1812.0	1812.0	1812.0	0.0
Tot:	2048.0		0.0	359.0	179.5	62.7	0.0	249.0	124.5	45.5	1789.0	1945.0	1867.0	31.2

n = 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1617.00)

	1.0	13/ 0	423.0	423.0	423.0	0.0	0.0	0.0	0.0	0.0	2040.0	2040.0	2040.0	0.0
	12.0	13/ 1	347.0	414.0	387.8	20.3	7.0	55.0	23.3	15.0	2006.0	2063.0	2028.0	16.6
	66.0	13/ 2	283.0	401.0	352.5	27.3	15.0	99.0	46.5	20.2	1972.0	2063.0	2016.0	22.4
	220.0	13/ 3	228.0	385.0	317.3	31.8	24.0	132.0	69.8	23.5	1940.0	2063.0	2004.0	26.0
	495.0	13/ 4	187.0	368.0	282.0	34.6	36.0	165.0	93.0	25.6	1911.0	2060.0	1992.0	28.3
	792.0	13/ 5	146.0	342.0	246.8	36.2	49.0	195.0	116.3	26.7	1899.0	2054.0	1980.0	29.6
	924.0	13/ 6	113.0	310.0	211.5	36.7	63.0	216.0	139.5	27.1	1890.0	2046.0	1968.0	30.1
	792.0	13/ 7	81.0	277.0	176.3	36.2	84.0	230.0	162.8	26.7	1882.0	2037.0	1956.0	29.6
	495.0	13/ 8	55.0	236.0	141.0	34.6	114.0	243.0	186.0	25.6	1876.0	2025.0	1944.0	28.3
	220.0	13/ 9	38.0	195.0	105.8	31.8	147.0	255.0	209.3	23.5	1873.0	2000.0	1932.0	26.0
	66.0	13/10	22.0	140.0	70.5	27.3	180.0	264.0	232.5	20.2	1873.0	1964.0	1920.0	22.4
	12.0	13/11	9.0	76.0	35.3	20.3	224.0	272.0	255.8	15.0	1873.0	1930.0	1908.0	16.6
	1.0	13/12	0.0	0.0	0.0	0.0	279.0	279.0	279.0	0.0	1896.0	1896.0	1896.0	0.0
Tot:	4096.0		0.0	423.0	211.5	70.4	0.0	279.0	139.5	47.9	1873.0	2063.0	1968.0	35.5

n = 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1827.00)

	1.0	14/ 0	460.0	460.0	460.0	0.0	0.0	0.0	0.0	0.0	2287.0	2287.0	2287.0	0.0
	13.0	14/ 1	384.0	451.0	424.6	19.5	5.0	55.0	21.8	15.2	2253.0	2310.0	2273.5	16.8
	78.0	14/ 2	320.0	438.0	389.2	26.4	12.0	99.0	43.7	20.6	2219.0	2310.0	2260.0	22.8
	286.0	14/ 3	265.0	422.0	353.8	30.8	20.0	132.0	65.5	24.0	2187.0	2310.0	2246.4	26.6
	715.0	14/ 4	224.0	405.0	318.5	33.7	29.0	165.0	87.4	26.3	2155.0	2307.0	2232.8	29.1
	1287.0	14/ 5	183.0	379.0	283.1	35.6	41.0	195.0	109.2	27.8	2126.0	2301.0	2219.3	30.7
	1716.0	14/ 6	146.0	347.0	247.7	36.5	54.0	216.0	131.1	28.4	2114.0	2293.0	2205.8	31.5
	1716.0	14/ 7	113.0	314.0	212.3	36.5	68.0	230.0	152.9	28.4	2105.0	2284.0	2192.2	31.5
	1287.0	14/ 8	81.0	277.0	176.9	35.6	89.0	243.0	174.8	27.8	2097.0	2272.0	2178.7	30.7
	715.0	14/ 9	55.0	236.0	141.5	33.7	119.0	255.0	196.6	26.3	2091.0	2243.0	2165.2	29.1
	286.0	14/10	38.0	195.0	106.2	30.8	152.0	264.0	218.5	24.0	2088.0	2211.0	2151.6	26.6
	78.0	14/11	22.0	140.0	70.8	26.4	185.0	272.0	240.3	20.6	2088.0	2179.0	2138.1	22.8
	13.0	14/12	9.0	76.0	35.4	19.5	229.0	279.0	262.2	15.2	2088.0	2145.0	2124.5	16.8
	1.0	14/13	0.0	0.0	0.0	0.0	284.0	284.0	284.0	0.0	2111.0	2111.0	2111.0	0.0
Tot:	8192.0		0.0	460.0	230.0	72.8	0.0	284.0	142.0	48.0	2088.0	2310.0	2199.0	38.9

n = 16 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 1927.00)

	1.0	15/ 0	492.0	492.0	492.0	0.0	0.0	0.0	0.0	0.0	2419.0	2419.0	2419.0	0.0
	13.0	15/ 1	416.0	483.0	456.6	19.5	5.0	55.0	21.8	15.2	2385.0	2442.0	2405.5	16.8
	78.0	15/ 2	352.0	470.0	421.2	26.4	12.0	99.0	43.7	20.6	2351.0	2442.0	2391.9	22.8
	286.0	15/ 3	297.0	454.0	385.8	30.8	20.0	132.0	65.5	24.0	2319.0	2442.0	2378.4	26.6
	715.0	15/ 4	256.0	437.0	350.5	33.7	29.0	165.0	87.4	26.3	2287.0	2439.0	2364.8	29.1
	1287.0	15/ 5	215.0	411.0	315.1	35.6	41.0	195.0	109.2	27.8	2258.0	2433.0	2351.3	30.7
	1716.0	15/ 6	178.0	379.0	279.7	36.5	54.0	216.0	131.1	28.4	2246.0	2425.0	2337.8	31.5
	1716.0	15/ 7	145.0	346.0	244.3	36.5	68.0	230.0	152.9	28.4	2237.0	2416.0	2324.2	31.5
	1287.0	15/ 8	113.0	309.0	208.9	35.6	89.0	243.0	174.8	27.8	2229.0	2404.0	2310.7	30.7
	715.0	15/ 9	87.0	268.0	173.5	33.7	119.0	255.0	196.6	26.3	2223.0	2375.0	2297.2	29.1
	286.0	15/10	70.0	227.0	138.2	30.8	152.0	264.0	218.5	24.0	2220.0	2343.0	2283.6	26.6
	78.0	15/11	54.0	172.0	102.8	26.4	185.0	272.0	240.3	20.6	2220.0	2311.0	2270.1	22.8
	13.0	15/12	41.0	108.0	67.4	19.5	229.0	279.0	262.2	15.2	2220.0	2277.0	2256.5	16.8
	1.0	15/13	32.0	32.0	32.0	0.0	284.0	284.0	284.0	0.0	2243.0	2243.0	2243.0	0.0
Tot:	8192.0		32.0	492.0	262.0	72.8	0.0	284.0	142.0	48.0	2220.0	2442.0	2331.0	38.9

n = 17 16 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2053.00)

	1.0	16/ 0	504.0	504.0	504.0	0.0	0.0	0.0	0.0	0.0	2557.0	2557.0	2557.0	0.0
	14.0	16/ 1	428.0	495.0	470.3	19.7	3.0	55.0	20.5	15.4	2523.0	2580.0	2543.8	16.3
	91.0	16/ 2	364.0	483.0	436.6	26.8	8.0	99.0	41.0	21.0	2489.0	2580.0	2530.6	22.1

364.0	16/ 3	309.0	470.0	402.9	31.4	15.0	132.0	61.5	24.6	2457.0	2580.0	2517.4	25.9
1001.0	16/ 4	268.0	454.0	369.1	34.6	23.0	165.0	82.0	27.1	2425.0	2577.0	2504.1	28.5
2002.0	16/ 5	227.0	437.0	335.4	36.7	32.0	195.0	102.5	28.7	2396.0	2571.0	2490.9	30.2
3003.0	16/ 6	190.0	411.0	301.7	37.9	44.0	216.0	123.0	29.7	2384.0	2563.0	2477.7	31.2
3432.0	16/ 7	157.0	379.0	268.0	38.3	57.0	230.0	143.5	30.0	2375.0	2554.0	2464.5	31.6
3003.0	16/ 8	125.0	346.0	234.3	37.9	71.0	243.0	164.0	29.7	2366.0	2545.0	2451.3	31.2
2002.0	16/ 9	99.0	309.0	200.6	36.7	92.0	255.0	184.5	28.7	2358.0	2533.0	2438.1	30.2
1001.0	16/10	82.0	268.0	166.9	34.6	122.0	264.0	205.0	27.1	2352.0	2504.0	2424.9	28.5
364.0	16/11	66.0	227.0	133.1	31.4	155.0	272.0	225.5	24.6	2349.0	2472.0	2411.6	25.9
91.0	16/12	53.0	172.0	99.4	26.8	188.0	279.0	246.0	21.0	2349.0	2440.0	2398.4	22.1
14.0	16/13	41.0	108.0	65.7	19.7	232.0	284.0	266.5	15.4	2349.0	2406.0	2385.2	16.3
1.0	16/14	32.0	32.0	32.0	0.0	287.0	287.0	287.0	0.0	2372.0	2372.0	2372.0	0.0
Tot: 16384.0		32.0	504.0	268.0	73.1	0.0	287.0	143.5	48.0	2349.0	2580.0	2464.5	39.2

n = 15 17 16 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2167.00)

1.0	17/ 0	530.0	530.0	530.0	0.0	0.0	0.0	0.0	0.0	2697.0	2697.0	2697.0	0.0
16.0	17/ 1	454.0	521.0	497.8	19.2	3.0	55.0	18.4	15.4	2663.0	2720.0	2683.2	15.6
120.0	17/ 2	390.0	518.0	465.6	26.4	6.0	99.0	36.9	21.1	2629.0	2720.0	2669.5	21.4
560.0	17/ 3	335.0	509.0	433.6	31.3	11.0	132.0	55.3	24.9	2597.0	2720.0	2655.9	25.4
1820.0	17/ 4	294.0	496.0	401.6	35.0	16.0	165.0	73.8	27.6	2565.0	2717.0	2642.4	28.2
4368.0	17/ 5	253.0	480.0	369.8	37.7	23.0	195.0	92.2	29.6	2536.0	2714.0	2628.9	30.3
8008.0	17/ 6	216.0	463.0	338.0	39.7	31.0	216.0	110.6	30.9	2509.0	2711.0	2615.6	31.8
11440.0	17/ 7	183.0	437.0	306.4	40.9	40.0	230.0	129.1	31.6	2497.0	2705.0	2602.4	32.7
12870.0	17/ 8	151.0	405.0	274.8	41.5	52.0	243.0	147.5	31.9	2488.0	2697.0	2589.3	33.1
11440.0	17/ 9	119.0	373.0	243.4	41.5	65.0	255.0	165.9	31.6	2479.0	2688.0	2576.3	33.0
8008.0	17/10	93.0	340.0	212.0	40.8	79.0	264.0	184.4	30.9	2471.0	2676.0	2563.4	32.3
4368.0	17/11	76.0	303.0	180.8	39.3	100.0	272.0	202.8	29.6	2465.0	2649.0	2550.6	31.1
1820.0	17/12	60.0	262.0	149.6	37.0	130.0	279.0	221.3	27.6	2462.0	2620.0	2537.9	29.2
560.0	17/13	47.0	221.0	118.6	33.5	163.0	284.0	239.7	24.9	2462.0	2588.0	2525.2	26.4
120.0	17/14	35.0	166.0	87.6	28.6	196.0	289.0	258.1	21.1	2462.0	2556.0	2512.7	22.5
16.0	17/15	26.0	102.0	56.8	21.1	240.0	292.0	276.6	15.4	2465.0	2522.0	2500.3	16.6
1.0	17/16	26.0	26.0	26.0	0.0	295.0	295.0	295.0	0.0	2488.0	2488.0	2488.0	0.0
Tot: 65536.0		26.0	530.0	275.0	74.7	0.0	295.0	147.5	48.1	2462.0	2720.0	2589.5	41.4

n = 8 15 17 16 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2207.00)

1.0	18/ 0	547.0	547.0	547.0	0.0	0.0	0.0	0.0	0.0	2754.0	2754.0	2754.0	0.0
18.0	18/ 1	471.0	538.0	515.9	18.5	3.0	55.0	17.2	15.0	2720.0	2777.0	2740.2	14.8
153.0	18/ 2	407.0	535.0	485.0	25.5	6.0	99.0	34.4	20.5	2686.0	2777.0	2726.4	20.4
816.0	18/ 3	352.0	526.0	454.1	30.3	11.0	132.0	51.7	24.3	2654.0	2777.0	2712.7	24.3
3060.0	18/ 4	311.0	513.0	423.2	34.0	16.0	165.0	68.9	27.1	2622.0	2774.0	2699.1	27.1
8568.0	18/ 5	270.0	497.0	392.5	36.9	23.0	195.0	86.1	29.2	2593.0	2771.0	2685.6	29.3
18564.0	18/ 6	233.0	480.0	361.8	39.0	30.0	216.0	103.3	30.8	2566.0	2768.0	2672.2	31.0
31824.0	18/ 7	200.0	463.0	331.3	40.6	38.0	230.0	120.6	31.8	2547.0	2762.0	2658.8	32.1
43758.0	18/ 8	168.0	437.0	300.8	41.6	46.0	243.0	137.8	32.4	2535.0	2754.0	2645.5	32.9
48620.0	18/ 9	136.0	411.0	270.3	42.1	55.0	255.0	155.0	32.6	2526.0	2745.0	2632.3	33.2
43758.0	18/10	110.0	379.0	240.0	42.0	67.0	264.0	172.2	32.4	2517.0	2736.0	2619.2	33.1
31824.0	18/11	84.0	347.0	209.7	41.5	80.0	272.0	189.4	31.8	2508.0	2724.0	2606.1	32.6
18564.0	18/12	67.0	314.0	179.5	40.3	94.0	280.0	206.7	30.8	2500.0	2705.0	2593.2	31.7
8568.0	18/13	50.0	277.0	149.4	38.5	115.0	287.0	223.9	29.2	2494.0	2678.0	2580.3	30.2
3060.0	18/14	34.0	236.0	119.4	36.0	145.0	294.0	241.1	27.1	2491.0	2649.0	2567.5	28.2
816.0	18/15	21.0	195.0	89.4	32.4	178.0	299.0	258.3	24.3	2491.0	2617.0	2554.7	25.3
153.0	18/16	9.0	140.0	59.5	27.5	211.0	304.0	275.6	20.5	2491.0	2585.0	2542.1	21.5
18.0	18/17	0.0	76.0	29.7	20.1	255.0	307.0	292.8	15.0	2494.0	2551.0	2529.5	15.7
1.0	18/18	0.0	0.0	0.0	0.0	310.0	310.0	310.0	0.0	2517.0	2517.0	2517.0	0.0
Tot: 262144.0		0.0	547.0	270.5	76.3	0.0	310.0	155.0	48.4	2491.0	2777.0	2632.5	42.7

n = 18 8 15 17 16 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2253.00)

1.0	19/ 0	554.0	554.0	554.0	0.0	0.0	0.0	0.0	0.0	2807.0	2807.0	2807.0	0.0
20.0	19/ 1	478.0	547.0	524.9	18.5	3.0	55.0	16.3	14.5	2773.0	2830.0	2794.1	14.4
190.0	19/ 2	414.0	542.0	495.9	25.7	6.0	99.0	32.5	20.0	2739.0	2830.0	2781.4	20.0
1140.0	19/ 3	359.0	535.0	467.0	30.8	11.0	132.0	48.8	23.7	2707.0	2830.0	2768.8	23.9
4845.0	19/ 4	318.0	526.0	438.3	34.8	16.0	165.0	65.0	26.6	2675.0	2830.0	2756.3	26.9
15504.0	19/ 5	277.0	518.0	409.8	38.0	23.0	195.0	81.3	28.8	2646.0	2829.0	2744.0	29.3
38760.0	19/ 6	240.0	509.0	381.4	40.5	30.0	216.0	97.5	30.5	2619.0	2829.0	2731.9	31.2

77520.0	19/ 7	207.0	496.0	353.2	42.5	37.0	230.0	113.8	31.7	2600.0	2826.0	2720.0	32.7
125970.0	19/ 8	175.0	480.0	325.1	44.0	45.0	243.0	130.0	32.6	2588.0	2823.0	2708.1	33.8
167960.0	19/ 9	143.0	463.0	297.1	45.1	53.0	255.0	146.3	33.1	2579.0	2820.0	2696.4	34.6
184756.0	19/10	117.0	437.0	269.4	45.7	61.0	264.0	162.5	33.2	2570.0	2814.0	2684.9	35.0
167960.0	19/11	91.0	411.0	241.7	45.8	70.0	272.0	178.8	33.1	2561.0	2806.0	2673.5	35.1
125970.0	19/12	74.0	379.0	214.3	45.5	82.0	280.0	195.0	32.6	2553.0	2797.0	2662.3	34.8
77520.0	19/13	57.0	347.0	187.0	44.7	95.0	288.0	211.3	31.7	2547.0	2785.0	2651.2	34.1
38760.0	19/14	41.0	314.0	159.8	43.3	109.0	295.0	227.5	30.5	2544.0	2766.0	2640.3	33.0
15504.0	19/15	28.0	277.0	132.8	41.2	130.0	302.0	243.8	28.8	2544.0	2739.0	2629.5	31.4
4845.0	19/16	16.0	236.0	105.9	38.4	160.0	309.0	260.0	26.6	2544.0	2710.0	2618.9	29.2
1140.0	19/17	7.0	195.0	79.2	34.5	193.0	314.0	276.3	23.7	2544.0	2678.0	2608.5	26.3
190.0	19/18	0.0	140.0	52.7	29.3	226.0	319.0	292.5	20.0	2547.0	2646.0	2598.2	22.2
20.0	19/19	0.0	76.0	26.3	21.4	270.0	322.0	308.8	14.5	2555.0	2612.0	2588.0	16.3
1.0	19/20	0.0	0.0	0.0	0.0	325.0	325.0	325.0	0.0	2578.0	2578.0	2578.0	0.0
Tot: 1048576.0		0.0	554.0	269.8	76.3	0.0	325.0	162.5	48.7	2544.0	2830.0	2685.3	42.7

n = 19 18 8 15 17 16 14 13 12 11 10 7 6 5 4 3 2 9 1 0 (Eval Cost :: 2349.00)

1.0	20/ 0	596.0	596.0	596.0	0.0	0.0	0.0	0.0	0.0	2945.0	2945.0	2945.0	0.0
22.0	20/ 1	520.0	589.0	565.7	18.0	3.0	55.0	15.5	14.0	2911.0	2968.0	2930.2	15.0
231.0	20/ 2	456.0	584.0	535.7	25.0	6.0	99.0	31.0	19.3	2877.0	2968.0	2915.7	20.7
1540.0	20/ 3	401.0	577.0	506.0	30.1	11.0	132.0	46.5	23.1	2843.0	2968.0	2901.5	24.7
7315.0	20/ 4	359.0	568.0	476.6	34.1	16.0	165.0	62.0	26.0	2811.0	2968.0	2887.6	27.7
26334.0	20/ 5	318.0	560.0	447.5	37.3	23.0	195.0	77.5	28.2	2779.0	2967.0	2874.0	30.1
74613.0	20/ 6	277.0	551.0	418.7	40.0	30.0	216.0	93.0	30.0	2750.0	2967.0	2860.7	32.0
170544.0	20/ 7	240.0	538.0	390.2	42.2	37.0	230.0	108.5	31.3	2723.0	2964.0	2847.7	33.5
319770.0	20/ 8	207.0	522.0	362.1	44.0	45.0	243.0	124.0	32.4	2704.0	2961.0	2835.1	34.7
497420.0	20/ 9	175.0	505.0	334.2	45.4	53.0	255.0	139.5	33.1	2692.0	2958.0	2822.7	35.6
646646.0	20/10	143.0	480.0	306.6	46.4	61.0	264.0	155.0	33.5	2683.0	2952.0	2810.6	36.2
705432.0	20/11	117.0	463.0	279.4	47.0	69.0	272.0	170.5	33.6	2674.0	2944.0	2798.9	36.5
646646.0	20/12	91.0	437.0	252.5	47.3	77.0	280.0	186.0	33.5	2665.0	2935.0	2787.5	36.5
497420.0	20/13	74.0	411.0	225.8	47.2	86.0	288.0	201.5	33.1	2657.0	2923.0	2776.3	36.2
319770.0	20/14	57.0	379.0	200.0	46.6	98.0	296.0	217.0	32.4	2651.0	2909.0	2765.5	35.7
170544.0	20/15	41.0	347.0	173.5	45.6	111.0	304.0	232.5	31.3	2648.0	2897.0	2755.0	34.8
74613.0	20/16	28.0	314.0	147.8	44.1	125.0	311.0	248.0	30.0	2648.0	2878.0	2744.8	33.5
26334.0	20/17	16.0	277.0	122.4	41.9	146.0	318.0	263.5	28.2	2648.0	2851.0	2734.9	31.8
7315.0	20/18	7.0	236.0	97.3	39.0	176.0	325.0	279.0	26.0	2648.0	2822.0	2725.3	29.5
1540.0	20/19	0.0	195.0	72.5	35.1	209.0	330.0	294.5	23.1	2651.0	2790.0	2716.0	26.5
231.0	20/20	0.0	140.0	48.0	29.7	242.0	335.0	310.0	19.3	2659.0	2758.0	2707.0	22.4
22.0	20/21	0.0	76.0	23.9	21.8	286.0	338.0	325.5	14.0	2667.0	2724.0	2698.4	16.4
1.0	20/22	0.0	0.0	0.0	0.0	341.0	341.0	341.0	0.0	2690.0	2690.0	2690.0	0.0
Tot: 4194304.0		0.0	596.0	280.3	78.4	0.0	341.0	170.5	49.0	2648.0	2968.0	2800.0	44.9

REFERENCES

[BERZ 91] Berzins, V., Luqi, *Software Engineering with Abstractions*, Addison-Wesley Publishing Co., 1991.

[BOLC 90] Bolchoz, J.M., *The Identification of Software Failure Regions*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1990.

[BOOCH 87] Booch, G., *Software Components with Ada*, The Benjamin/Cummings Publishing Company Inc., 1987.

[CAMU 90] Camuffo, M., Maiocchi, M., Morselli, M., *"Automatic Software Test Generation"* .

[DEMI 87] Demillo, R.A., McCracken, W.M., Martin, R.J., Passafiume, J.F., *Software Testing and Evaluation*, The Benjamin/Cummings Publishing Company Inc., 1987

[DINS 87] Dinsmore G., Kane R., *Battle Simulation-Test Precis*, Unpublished Manuscript, University of California, Irvine, CA, 1987.

[HETZ 88] Hetzel, W.C., *The Complete Guide to Software Testing*, QED Information Services Inc., 1988.

[IEEE 91] IEEE *Software Engineering Standards Collection*, The Institute of Electrical and Electronics Engineers, Inc., 1991.

[MYER 79] Myers, G.J., The Art of Software Testing, John Wiley & Sons, Inc., 1979.

[NGUY 88] Nguyen, T., Forester, K., *AFLEX - An Ada Lexical Analysis Generator, Version 1.0*, Arcadia Environment Research Project, Department of Information and Computer Science, University of California, Irvine, 1988.

[SHIM 87] Shimeall, T.J., Lief, S., Dinsmore G., Kane R., *Test Variant of the Battle Simulation Specification*, Unpublished Manuscript, University of California, Irvine, CA, 1987.

[SHIM 89] Shimeall, T.J., *An Empircal Comparison of Software Fault Tolerance and Fault Elimination*, PHD Dissertation, University of California, Irvine, CA, 1989.

[SHIM 90] Shimeall, T.J., *CONFLICT Specification*, Naval Postgraduate School, Monterey, CA, October 1990.

[TABA 88] Taback, D., Deepak, T., *AYACC Users Manual, Version 1.0*, Arcadia Environment Research Project, Department of Information and Computer Science, University of California, Irvine, 1988.

INITIAL DISTRIBUTION LIST

Defense Technical Information Center Cameron Station Alexandria, VA 221314	2
Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943	2
Chairman, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	2
Dr. Timothy J. Shimeall Computer Science Department.Code CSSm Naval Postgraduate School Monterey, CA 93943	6
Dr. Man-Tak Shing Computer Science Department.Code CSSh Naval Postgraduate School Monterey, CA 93943	1
LT Timothy J. Kelly 745 Lamplight Ln Hazelwood, MO 63042	2

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY CA 93943-5101

AMERICAN BINDERY
6-94
2

DUDLEY KNOX LIBRARY
3 2768 00019398 1